Linear Models of Classification: Probabilistic Generative and Discriminative Models

Prof. Nicholas Zabaras
Center for Informatics and Computational Science
https://cics.nd.edu/
University of Notre Dame
Notre Dame, Indiana, USA

Email: nzabaras@gmail.com
URL: https://www.zabaras.com/

February 19, 2019

Contents

- Linear models for classification, Approaches to Classification, Discriminant Functions, Least Squares approach to Classification
- Generative Vs Discriminative Classifiers, Fishers linear discriminant, Probabilistic Interpretation
- Probabilistic Generative Models for two Classes, Logistic Sigmoid, Models with K>2, Gaussian Class Conditionals, MLE, Multiclass Case, Discrete Features, Generalization to Exponential Family Class Conditionals
- Online Learning and Stochastic Optimization – Robbins-Monro Algorithm, The Perceptron algorithm, Bayesian perspective
- Probabilistic Discriminative Models, Nonlinear Basis in Classification Models, Logistic Regression and Generalized Linear Models, Cross Entropy Error, Sequential Update, Linearly Separable Data, Regularization, Iterative Reweighted Least Squares, Multiclass Logistic Regression
- Algorithms for Uncostrained Optimization, Steepest Descent Method, Newton's Method, BFGS, Regularization
- Bayesian Logistic Regression, Laplace Approximation, Model comparison and Model Evidence, BIC Criterion, Gaussian Approximation to the Posterior, Predictive Distribution, Probit Approximation, Outlier Detection
- Probit Regression, IRLS for Probit Regression, Canonical Link Functions

- Following closely Chris Bishops' PRML book, Chapter 4.
- K. Murphy, Machine Learning: A probabilistic Perspective, Chapter 8

Linear Models for Classification

- The goal in classification is to take a D −dimensional input $\boldsymbol{x}$ and to assign it to one of K discrete classes $C_k, k = 1, \ldots, K$.

- Decision Regions and Decision Boundaries: The input space is divided into decision regions whose boundaries are called decision boundaries.

- Linear Models for Classification: The decision boundaries are linear functions of $\boldsymbol{x}$ defined by $(D - 1)$ −dimensional hyperplanes within the D −dimensional input space.

- Linearly Separable Data Sets: Data sets whose classes can be separated exactly by linear decision surfaces are said to be linearly separable.

Classification: Target Values

- There are many ways of using target values to represent class labels.

- For probabilistic 2 class models, there is a single target variable $t \in \{0,1\}$. $t = 1$ represents class C_1, $t = 0$ class C_2.

  - We interpret the value of *t* as the probability that the class is C_1.

- For $K > 2$ classes, we use a *1-of-K* coding in which $\boldsymbol{t}$ is a vector of length K such that if the class is C_j, then all elements t_k *of* $\boldsymbol{t}$ are zero except $t_j = 1$.

  - E.g., for $K = 5$, then a pattern from class 2 has the target

$$\boldsymbol{t} = (0, 1, 0, 0, 0)^T$$

  - We interpret the value of t_k as the probability that the class is C_k.

Approaches to Classification

- Case I: Construct a discriminant function that directly assigns each vector $\boldsymbol{x}$ to a specific class.
- Case II: Model the conditional probability distribution $p(C_k|\boldsymbol{x})$ in an inference stage, and subsequently use this to make optimal decisions.
  - Case IIa: Model $p(C_k|\boldsymbol{x})$ directly by parametric models and then optimize the parameters using a training set.
  - Case IIb: Use a generative approach in which we model the class-conditional densities $p(\boldsymbol{x}|C_k)$, together with the priors $p(C_k)$ for the classes, and then compute $p(C_k|\boldsymbol{x})$ using Bayes' theorem

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{p(\boldsymbol{x})}$$

Discriminant Functions

Generalized Linear Models of Classification

- To predict discrete class labels e.g. posterior probabilities in the range $(0,1)$, we consider *a generalization of the* Bayesian regression model by transforming a linear function of $\boldsymbol{w}$ with a nonlinear function $f(\cdot)$:

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^T \boldsymbol{x} + w_0\right)$$

- $f(\cdot)$ is known as the activation function.

- The decision surfaces correspond to $y(\boldsymbol{x}) = \text{constant}$, so that $\boldsymbol{w}^T \boldsymbol{x} + w_0 = const.$, i.e. the decision surfaces are linear in $\boldsymbol{x}$, even if $f(\cdot)$ is nonlinear.

- These models are called *generalized linear models.*

- McCullagh, P. and J. Nelder (1989). *Generalized linear models*. Chapman and Hall. 2nd edition.

Generalized Linear Models of Classification

- Note that the generalized linear models of classification

$$y(\boldsymbol{x}) = f\left(\boldsymbol{w}^T \boldsymbol{x} + w_0\right)$$

  are not linear in $\boldsymbol{w}$ as is the case in regression models.

- As a result, classifications models are more complex analytically and computationally.

- The classification techniques can take place
  - directly in the input space $\boldsymbol{x}$ or
  - in a feature space defined by the basis functions $\boldsymbol{\phi}(\boldsymbol{x})$.

Discriminant Functions

- A discriminant is a function that takes $\boldsymbol{x}$ and assigns it directly to one of the K classes C_k.

- We consider here linear discriminants for which the decision surfaces are hyperplanes.

- For two classes, a linear discriminant function is of the form:

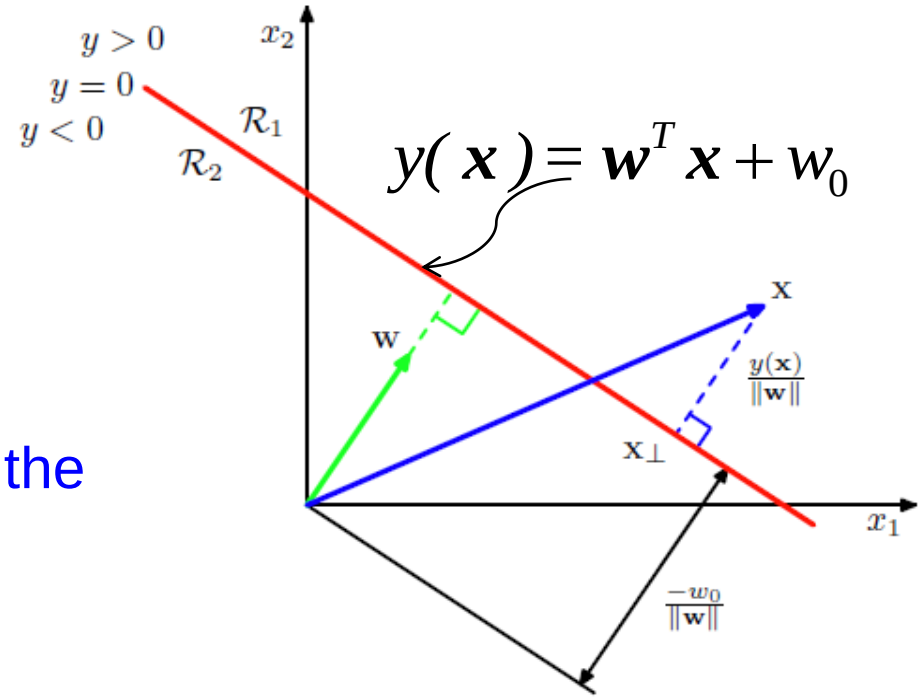
$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

$\boldsymbol{w}$ is the weight vector, w_0 the bias ($-w_0$ is called the threshold).

- We consider next the case of two classes.

Discriminant Functions For Two Classes

- $\boldsymbol{x}$ is assigned to C_1 if $y(\boldsymbol{x}) \geq 0$ and to C_2 otherwise.
- The decision boundary is $y(\boldsymbol{x}) = 0$, a $(D-1)$ –dimensional hyperplane within the D –dimensional input space.
- Consider two points $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ on the decision surface. Because $y(\boldsymbol{x}_A) = y(\boldsymbol{x}_B) = 0$, we have $\boldsymbol{w}^T(\boldsymbol{x}_A - \boldsymbol{x}_B) = 0$ and hence $\boldsymbol{w}$ is orthogonal to every vector on the decision surface.
- $\boldsymbol{w}$ determines the orientation of the decision surface.

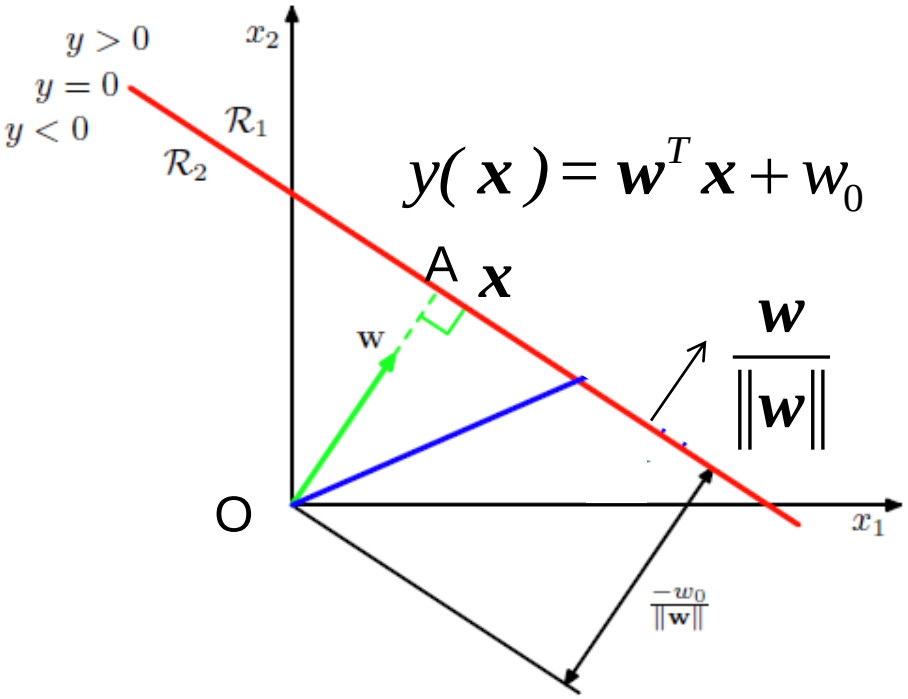


Discriminant Functions For Two Classes

- Consider the normal distance (OA) from the origin to the decision surface. Let $\boldsymbol{x}$ the projection of the origin on the decision surface. Since $y(\boldsymbol{x}) = 0$,

$$\overrightarrow{OA} = \boldsymbol{x} = \alpha \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \underset{\boldsymbol{w}^T \boldsymbol{x} + w_0 = 0}{\Rightarrow} \boldsymbol{w}^T \alpha \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} + w_0 = 0 \Rightarrow (OA) = \alpha = -\frac{w_0}{\|\boldsymbol{w}\|}$$

- The bias parameter w_0 determines the location of the decision surface from the origin.



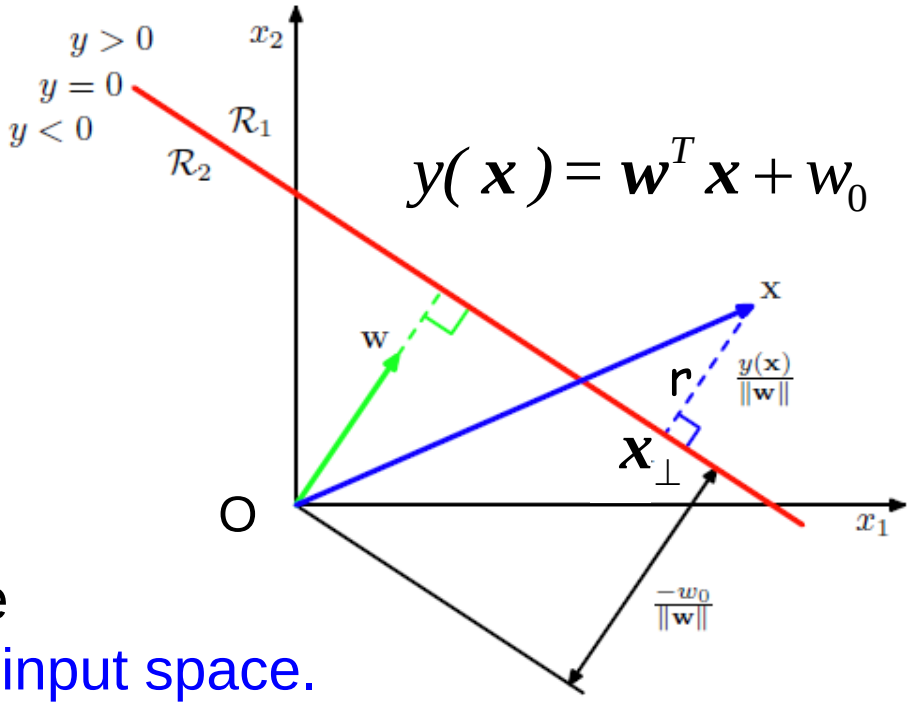
Discriminant Functions For Two Classes

- We note that the value of $y(\boldsymbol{x})$ for any $\boldsymbol{x}$ gives a signed measure of the distance r of $\boldsymbol{x}$ from the decision surface.

$$\boldsymbol{x} = \boldsymbol{x}_\perp + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \Rightarrow \underbrace{\boldsymbol{w}^T\boldsymbol{x} + w_0}_{y(\boldsymbol{x})} = \underbrace{\boldsymbol{w}^T\boldsymbol{x}_\perp + w_0}_{=0} + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} = r\|\boldsymbol{w}\| \Rightarrow r = \frac{y(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

- We can use compact notation and introduce $x_0 = 1$ and

$$\tilde{\boldsymbol{w}} = (w_0, \boldsymbol{w}),\ \tilde{\boldsymbol{x}} = (x_0, \boldsymbol{x})$$
$$y(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{w}}^T\boldsymbol{x}$$

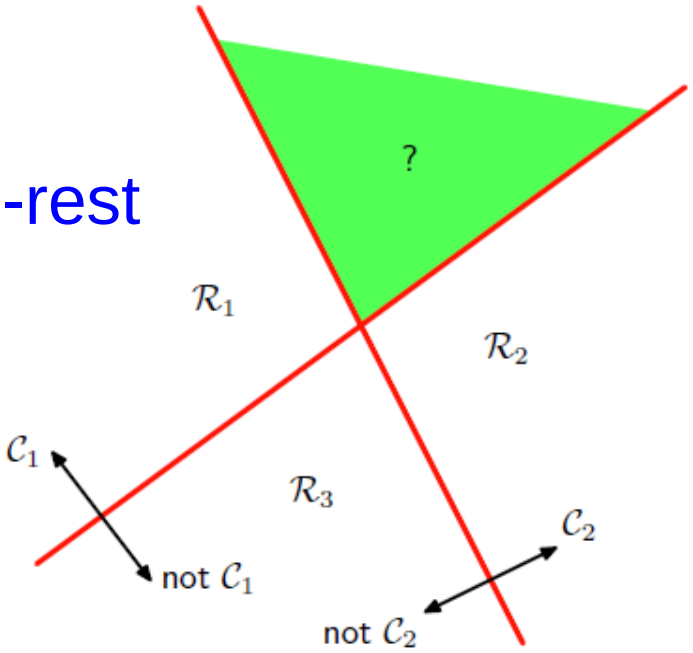


- The decision surfaces are now D −dimensional hyperplanes passing through the origin of the $D + 1$ −dimensional expanded input space.

Discriminant Functions For Many Classes

- Building a K −class discriminant by combining a number of two-class discriminant functions does not work in general.
- Consider the use of $(K-1)$ classifiers each of which solves a two-class problem of separating points in a particular class C_k from points not in that class.

one-versus-the-rest classifier

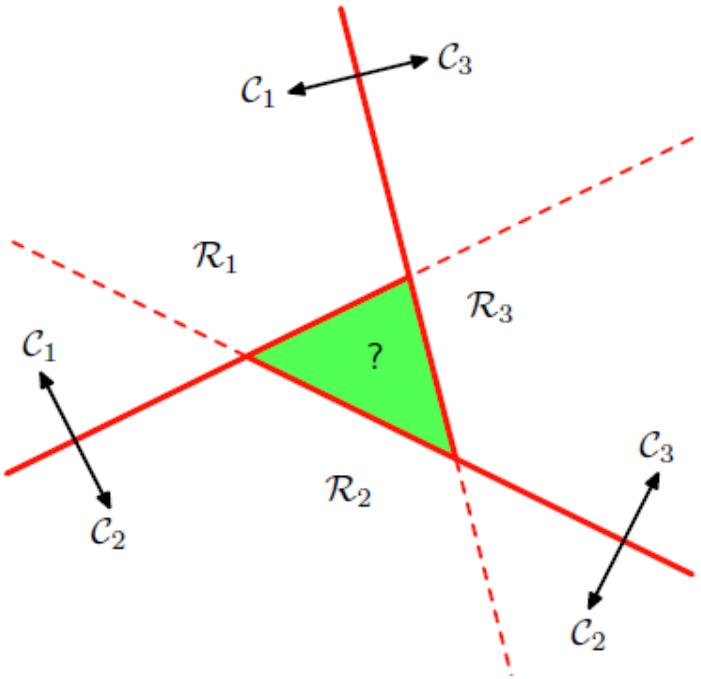


- Duda, R. O., P. E. Hart, and D. G. Stork (2001). *Pattern Classification*. Wiley Interscience. 2nd edition.

Discriminant Functions For Many Classes

- One can introduce $K(K-1)/2$ binary discriminant functions, *one for every possible pair of classes*.
- Each point is classified using a majority vote amongst the discriminant functions.
- This also leads to ambiguous regions.

one-versus-one classifier



Discriminant Functions For Many Classes

- To resolve the problems of the earlier classifiers, we consider a single K −class discriminant comprising K linear functions of the form

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

- We assign $\boldsymbol{x}$ to class C_k *if* $y_k(\boldsymbol{x}) > y_j(\boldsymbol{x})$ *for all* $j \neq k$.

- The decision boundary between class C_k *and class* C_j is therefore given *by* $y_k(\boldsymbol{x}) = y_j(\boldsymbol{x})$. It is a $(D - 1)$ −dimensional hyperplane

$$\left(\boldsymbol{w}_k - \boldsymbol{w}_j\right)^T \boldsymbol{x} + w_{k0} - w_{j0} = 0$$

- It has the same form as the decision boundary for the two-class case.

Discriminant Functions For Many Classes

- The decision regions of this discriminant are singly connected and convex.
- Consider $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ which lie inside decision region $\mathcal{R}_k$. Any $\boldsymbol{x}$ on the line connecting $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ can be expressed as

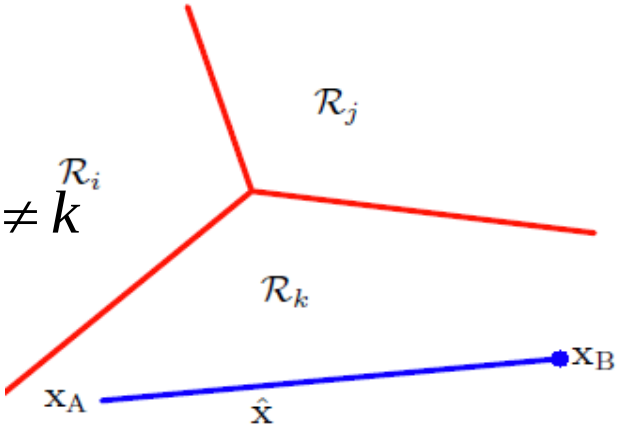
$$\hat{\boldsymbol{x}} = \lambda \boldsymbol{x}_A + (1-\lambda)\boldsymbol{x}_B, \quad 0 \le \lambda \le 1$$

- From the linearity of the discriminant functions we have:

$$y_k(\hat{\boldsymbol{x}}) = \lambda y_k(\boldsymbol{x}_A) + (1-\lambda) y_k(\boldsymbol{x}_B)$$

- Using $y_k(\boldsymbol{x}_A) > y_j(\boldsymbol{x}_A), y_k(\boldsymbol{x}_B) > y_j(\boldsymbol{x}_B), j \neq k$

  we see: $y_k(\hat{\boldsymbol{x}}) > y_j(\hat{\boldsymbol{x}}), j \neq k$ and so $\hat{\boldsymbol{x}} \in \mathcal{R}_k$.



- For 2 classes, one can use 2 discriminants or a single one as discussed earlier.

Least Squares for Classification

Least Squares Formalism to Classification

- Consider a general classification problem with K classes, with a *1-of-K* binary coding scheme for the target $\boldsymbol{t}$.

- One justification for *the least squares approach is that it approximates the conditional expectation* $\mathbb{E}[\boldsymbol{t}|\boldsymbol{x}]$.

  - For the binary coding scheme, $\mathbb{E}[\boldsymbol{t}|\boldsymbol{x}]$ is given by the vector of posterior class probabilities.

  - The approximation of these posterior class probabilities is poor and can have values outside the range $(0, 1)$.

Least Squares Formalism to Classification

- Each class is defined by a linear model:

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}, \; k = 1, \ldots, K$$

- We put all these equations together in a vector form:

$$\boldsymbol{y}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T \, \widetilde{\boldsymbol{x}}$$

- The k^{th} column of $\widetilde{\boldsymbol{W}}$ $(D+1 \times K)$ comprises the D + 1-dim vector $\widetilde{\boldsymbol{w}}_k = \left(w_{k0}, \boldsymbol{w}_k^T\right)^T$ and $\widetilde{\boldsymbol{x}} = (1, \boldsymbol{x}^T)^T$.

$$\widetilde{\boldsymbol{W}} = \begin{pmatrix} \uparrow \\ \widetilde{\boldsymbol{w}}_k \\ \downarrow \end{pmatrix}$$

- A new input $\boldsymbol{x}$ is then assigned to the class for which the output $y_k(\boldsymbol{x}) = \widetilde{\boldsymbol{w}}_k^T \widetilde{\boldsymbol{x}}$ is the largest.

Least Squares Formalism to Classification

- Consider a training data set: $\{\boldsymbol{x}_n, \boldsymbol{t}_n\}, n = 1, \ldots, N.$

$$y_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}, k = 1, \ldots, K \quad \text{or} \quad \boldsymbol{y}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T \tilde{\boldsymbol{x}}$$

- We define matrices $\widetilde{\boldsymbol{W}}, \boldsymbol{T}, \widetilde{\boldsymbol{X}}$ as follows:

$$\widetilde{\boldsymbol{X}} = \underset{N\times(D+1)}{\begin{pmatrix} \longleftarrow\ \tilde{\boldsymbol{x}}_{\boldsymbol{n}}^{\boldsymbol{T}}\ \longrightarrow \end{pmatrix}} \text{n}^{\text{th}}\text{ row} \quad \widetilde{\boldsymbol{W}} = \underset{(D+1)\times K}{\begin{pmatrix} \uparrow \\ \tilde{\boldsymbol{w}}_k \\ \downarrow \end{pmatrix}} \quad \boldsymbol{T} = \underset{N\times K}{\begin{pmatrix} \longleftarrow\ \boldsymbol{t}_n^T\ \longrightarrow \end{pmatrix}} \text{n}^{\text{th}}\text{ row}$$

- We can now write the sum-of-squares error and minimize:

$$E_D(\widetilde{\boldsymbol{W}}) = \frac{1}{2} Tr\left\{\left(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T}\right)^T \left(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T}\right)\right\} \Rightarrow \widetilde{\boldsymbol{W}} = \left(\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}}\right)^{-1}\widetilde{\boldsymbol{X}}^T\boldsymbol{T} = \widetilde{\boldsymbol{X}}^{\dagger}\boldsymbol{T}$$

$$\text{Note}: tr\left(\boldsymbol{A}^T\boldsymbol{A}\right) = \sum_{i,j} A_{ij}^2$$

Least Squares Formalism to Classification

- For a new input $\boldsymbol{x}$, we obtain the following discriminant function:

$$\boldsymbol{y}(\boldsymbol{x}) = \widetilde{\boldsymbol{W}}^T \widetilde{\boldsymbol{x}} = \boldsymbol{T}^T \left(\widetilde{\boldsymbol{X}}^\dagger\right)^T \widetilde{\boldsymbol{x}}$$

- Note that if every target vector in the training set satisfies some linear constraint $\boldsymbol{a}^T \boldsymbol{t}_n + b = 0$ for some $\boldsymbol{a}, b,$ then the model prediction for any $\boldsymbol{x}$ satisfies (see proof here)

$$\boldsymbol{a}^T \boldsymbol{y}(\boldsymbol{x}) + b = 0$$

  - If we use a 1-of-K coding scheme for K classes, the elements of $\boldsymbol{y}(\boldsymbol{x})$ will sum to 1 for any $\boldsymbol{x}$.
  - However, this constraint is not sufficient to allow the model outputs to be interpreted as probabilities since they are not constrained to lie within $(0, 1)$.

Appendix: Least Squares For Classification

- We can re-write the earlier equations identifying the bias:

$$E_D(\widetilde{\boldsymbol{W}}) = \frac{1}{2} Tr\left\{\left(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T}\right)^T \left(\widetilde{\boldsymbol{X}}\widetilde{\boldsymbol{W}} - \boldsymbol{T}\right)\right\} = \frac{1}{2} Tr\{(\boldsymbol{X}\boldsymbol{W} + \mathbf{1}\boldsymbol{w}_0^T - \boldsymbol{T})^T (\boldsymbol{X}\boldsymbol{W} + \mathbf{1}\boldsymbol{w}_0^T - \boldsymbol{T})\}$$

- $\boldsymbol{w}_0$ is the column vector of the bias weights (top row transposed of $\widetilde{\boldsymbol{W}}$). Differentiation wrt $\boldsymbol{w}_0$ → a familiar result:*

$$2N\boldsymbol{w}_0 + 2(\boldsymbol{X}\boldsymbol{W} - \boldsymbol{T})^T \mathbf{1} \Rightarrow \boldsymbol{w}_0 = \bar{\boldsymbol{t}} - \boldsymbol{W}^T \bar{\boldsymbol{x}}, \;\; \bar{\boldsymbol{t}} = \frac{1}{N}\boldsymbol{T}^T \mathbf{1}, \;\; \bar{\boldsymbol{x}} = \frac{1}{N}\boldsymbol{X}^T \mathbf{1}$$

- The error function now takes the form:

$$E_D(\boldsymbol{W}) = \frac{1}{2} Tr\{(\boldsymbol{X}\boldsymbol{W} + \overline{\boldsymbol{T}} - \overline{\boldsymbol{X}}\boldsymbol{W} - \boldsymbol{T})^T (\boldsymbol{X}\boldsymbol{W} + \overline{\boldsymbol{T}} - \overline{\boldsymbol{X}}\boldsymbol{W} - \boldsymbol{T})\}, \; \overline{\boldsymbol{T}} = \mathbf{1}\bar{\boldsymbol{t}}^T, \; \overline{\boldsymbol{X}} = \mathbf{1}\bar{\boldsymbol{x}}^T$$

- Setting the derivative wrt $\boldsymbol{W}$ equal to zero gives:

$$\boldsymbol{W} = \left(\widehat{\boldsymbol{X}}^T \widehat{\boldsymbol{X}}\right)^{-1} \widehat{\boldsymbol{X}}^T \widehat{\boldsymbol{T}} = \widehat{\boldsymbol{X}}^{\dagger} \widehat{\boldsymbol{T}}, \;\; \widehat{\boldsymbol{X}} = \boldsymbol{X} - \overline{\boldsymbol{X}}, \;\; \widehat{\boldsymbol{T}} = \boldsymbol{T} - \overline{\boldsymbol{T}}$$

* Use the identities $\frac{\partial}{\partial \boldsymbol{a}} Tr(\boldsymbol{a}\boldsymbol{a}^T) = \boldsymbol{a}$ and $\frac{\partial}{\partial \boldsymbol{a}} Tr(\boldsymbol{a}\boldsymbol{b}^T) = \boldsymbol{b}$. Thus e.g. $\frac{\partial}{\partial \boldsymbol{w}_0} Tr\left((1\boldsymbol{w}_0^T)^T (1\boldsymbol{w}_0^T)\right) = N \frac{\partial}{\partial \boldsymbol{w}_0} Tr(\boldsymbol{w}_0 \boldsymbol{w}_0^T) = N\boldsymbol{w}_0$.

Appendix: Least Squares For Classification

- For a new input $\boldsymbol{x}^*$, the prediction is:

$$\boldsymbol{y}(\boldsymbol{x}^*) = \boldsymbol{W}^T \boldsymbol{x}^* + \boldsymbol{w}_0 = \boldsymbol{W}^T \boldsymbol{x}^* + \bar{\boldsymbol{t}} - \boldsymbol{W}^T \bar{\boldsymbol{x}} = \bar{\boldsymbol{t}} - \widehat{\boldsymbol{T}}^T \left(\widehat{\boldsymbol{X}}^\dagger\right)^T (\boldsymbol{x}^* - \bar{\boldsymbol{x}})$$

- Consider for some $\boldsymbol{a}$, b, $\boldsymbol{a}^T \boldsymbol{t}_n + b = 0$. Apply this to $\bar{\boldsymbol{t}} = \frac{1}{N} \boldsymbol{T}^T \boldsymbol{1}$

$$\boldsymbol{a}^T \bar{\boldsymbol{t}} = \frac{1}{N} \boldsymbol{a}^T \boldsymbol{T}^T \boldsymbol{1} = -b \qquad \boldsymbol{T} = \begin{pmatrix} \longleftarrow \boldsymbol{t}_n^T \longrightarrow \end{pmatrix}_{N \times K}$$

- Applying $\boldsymbol{a}^T \boldsymbol{t}_n + b = 0$ to $\boldsymbol{y}(\boldsymbol{x}^*) = \bar{\boldsymbol{t}} - \widehat{\boldsymbol{T}}^T \left(\widehat{\boldsymbol{X}}^\dagger\right)^T (\boldsymbol{x}^* - \bar{\boldsymbol{x}})$ we obtain:

$$\boldsymbol{a}^T \boldsymbol{y}(\boldsymbol{x}^*) = \boldsymbol{a}^T \bar{\boldsymbol{t}} - \boldsymbol{a}^T \widehat{\boldsymbol{T}}^T \left(\widehat{\boldsymbol{X}}^\dagger\right)^T (\boldsymbol{x}^* - \bar{\boldsymbol{x}}) = \boldsymbol{a}^T \bar{\boldsymbol{t}} = -\mathrm{b}$$

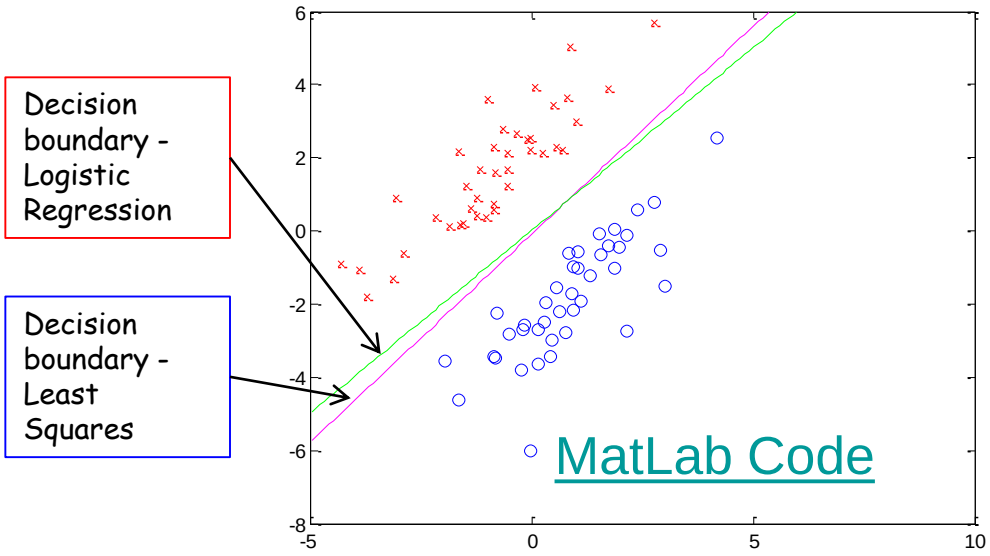
where $\boldsymbol{a}^T \widehat{\boldsymbol{T}}^T = \boldsymbol{a}^T (\boldsymbol{T} - \overline{\boldsymbol{T}})^T = \boldsymbol{a}^T \boldsymbol{T}^T - \boldsymbol{a}^T \bar{\boldsymbol{t}} \boldsymbol{1}^T = -b \boldsymbol{1}^T + b \boldsymbol{1}^T = \boldsymbol{0}^T$

- This is the proof of the earlier remark.

Least Squares Formalism to Classification

- The discriminant function : $y(x) = \widetilde{W}^T \tilde{x} = T^T \left(\tilde{X}^\dagger\right)^T \tilde{x}$ can be used to make decisions directly without any probabilistic interpretation.
- The least-squares solution in the example below defines two activation functions.
  - The decision boundary (for least squares) shown is taken as:

$$\left(w_1 - w_2\right)^T x + w_{10} - w_{20} = 0$$



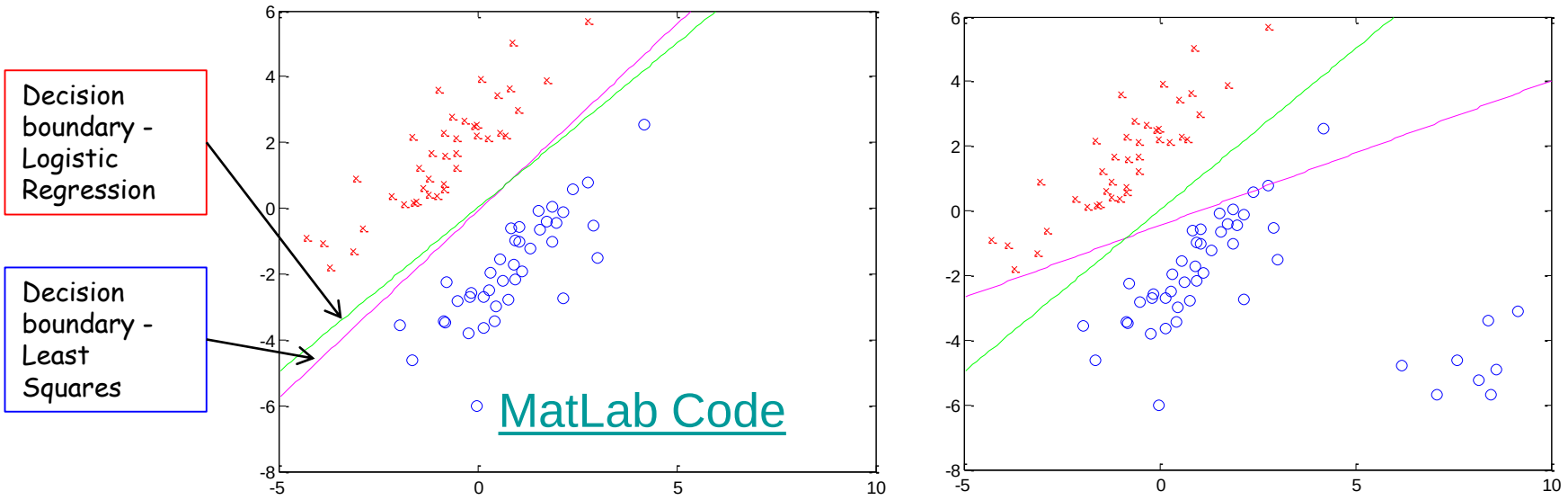
MatLab Code

- We will see in a follow up lecture that the decision boundary for the $2-$ class logistic regression algorithm in the input space x is defined as

$$p\left(C_1 \mid x\right) = \sigma\left(w^T x\right) = 0.5 \Rightarrow w^T x = 0$$

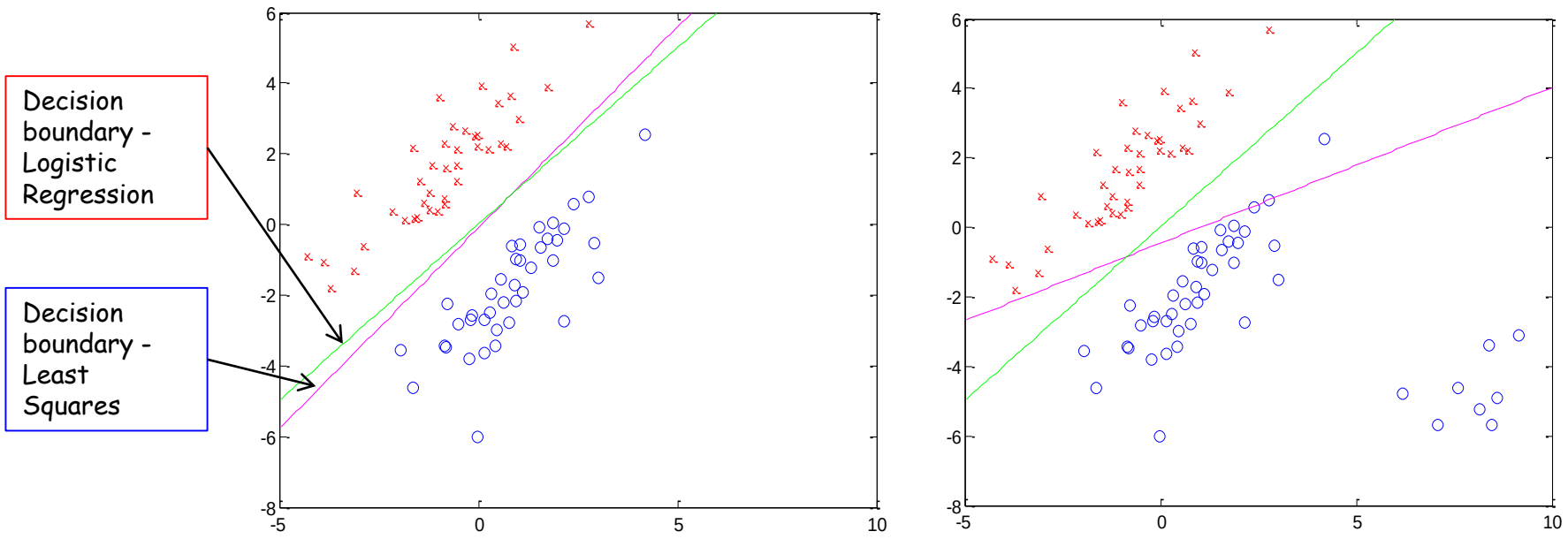
Least Squares Formalism to Classification

- As is the case for regression problems, the least-squares solution lacks robustness to outliers (see Fig).
  - The sum-of-squares error function penalizes predictions that are 'too correct' in that they lie a long way on the correct side of the decision boundary.



Least Squares is Sensitive to Outliers

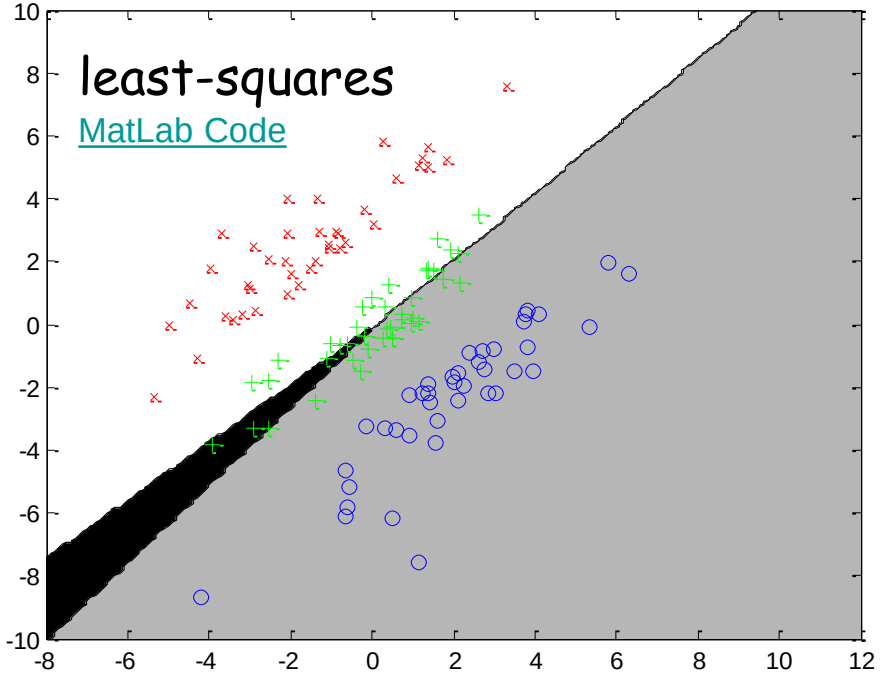
- Additional data points (right) produce a significant change in the location of the decision boundary (compare with the results from logistic regression in the input space $\boldsymbol{x}$ to be discussed in a coming lecture)
  - Note that these points would be correctly classified by the original decision boundary on the left.

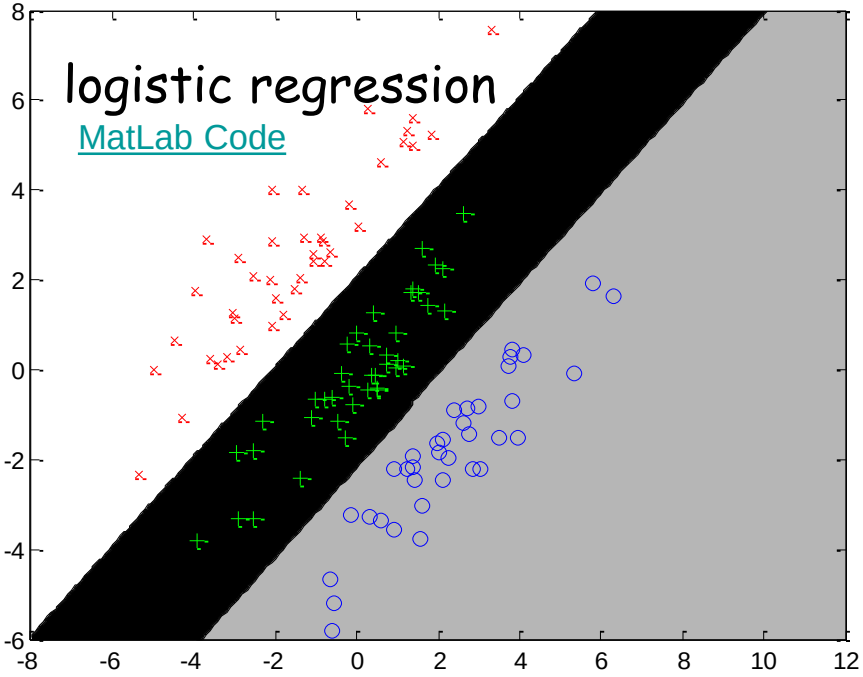


MatLab Code

Least Squares for Multiclass Classification

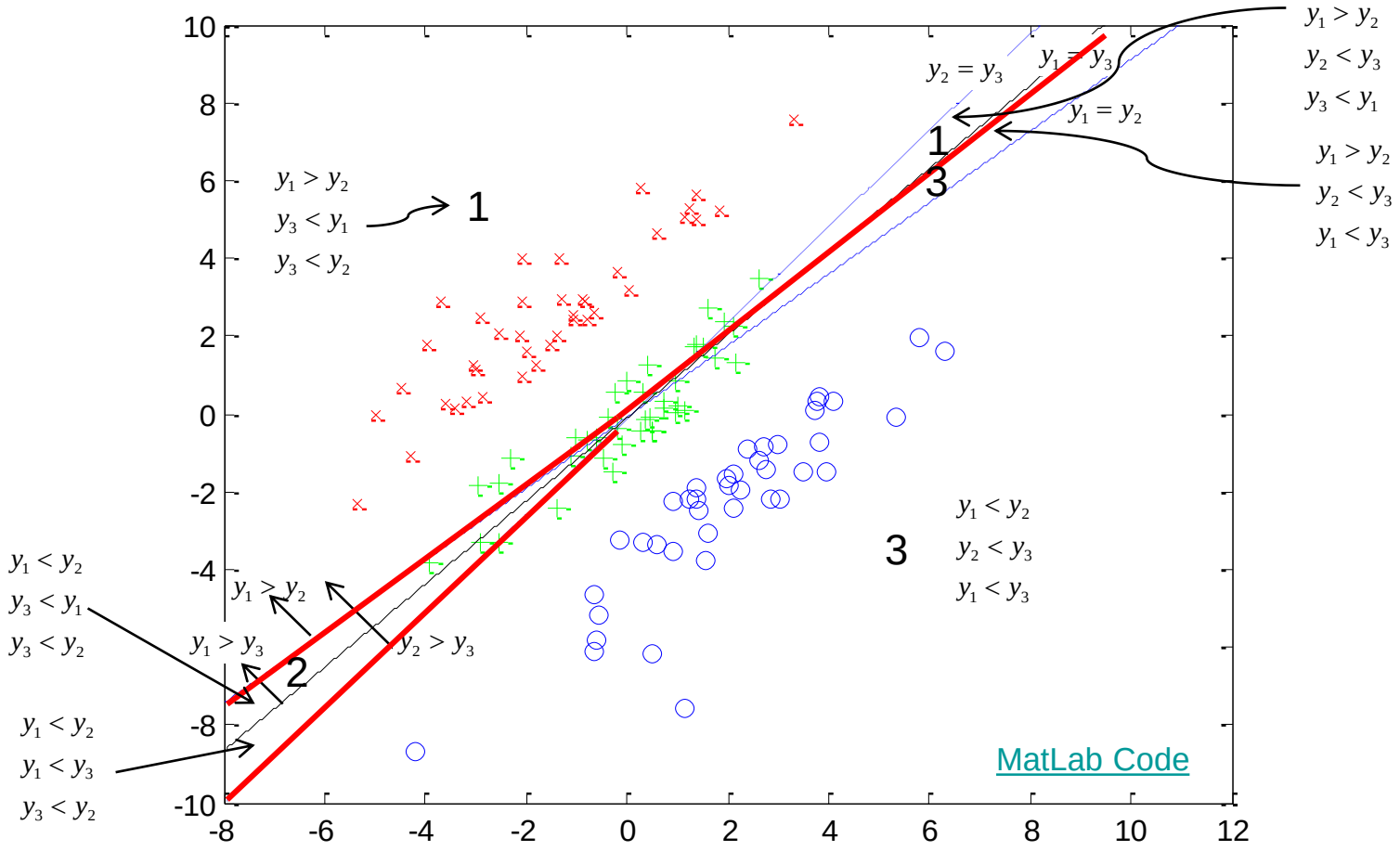
- Here we consider three classes.
- Three activation functions are defined and for each point, class k is assigned if $y_k(x) > y_j(x)$ for all $j \neq k$.
- For logistic regression, the regions are assigned based on posterior probabilities $p(C_m|\boldsymbol{\phi}) = \max\left(p(C_1|\boldsymbol{\phi}), p(C_2|\boldsymbol{\phi}), p(C_3|\boldsymbol{\phi})\right)$ with $p\left(C_k|\boldsymbol{\phi}\right) = \exp\left(a_k\right) \Big/ \sum_j \exp\left(a_j\right), a_k = w_k^T \boldsymbol{\phi}$





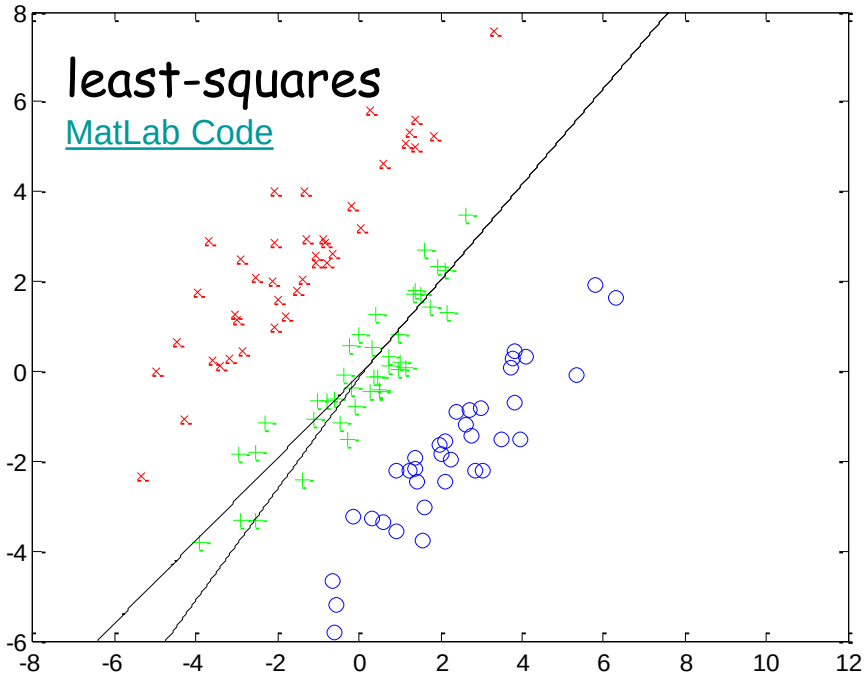
Decision Boundaries for the K Class Case

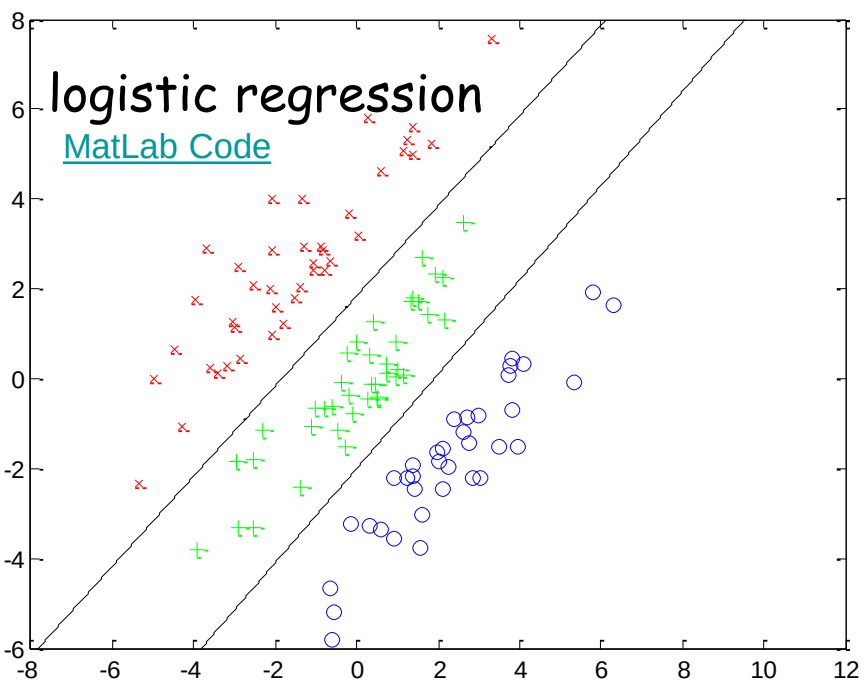
- For the LS case, we show using the 3 activation functions the potential decision boundaries. A region is assigned to class j when $y_j(x) > y_k(x)$ for all $k \neq j$. Verify the class assignments.



Least Squares for Multiclass Classification

- In this example of 3 classes, logistic regression gives a good solution.
- The least-squares solution gives poor results. Least squares correspond to MLE under the assumption of a Gaussian conditional distribution, whereas binary target vectors have a distribution that is far from Gaussian.



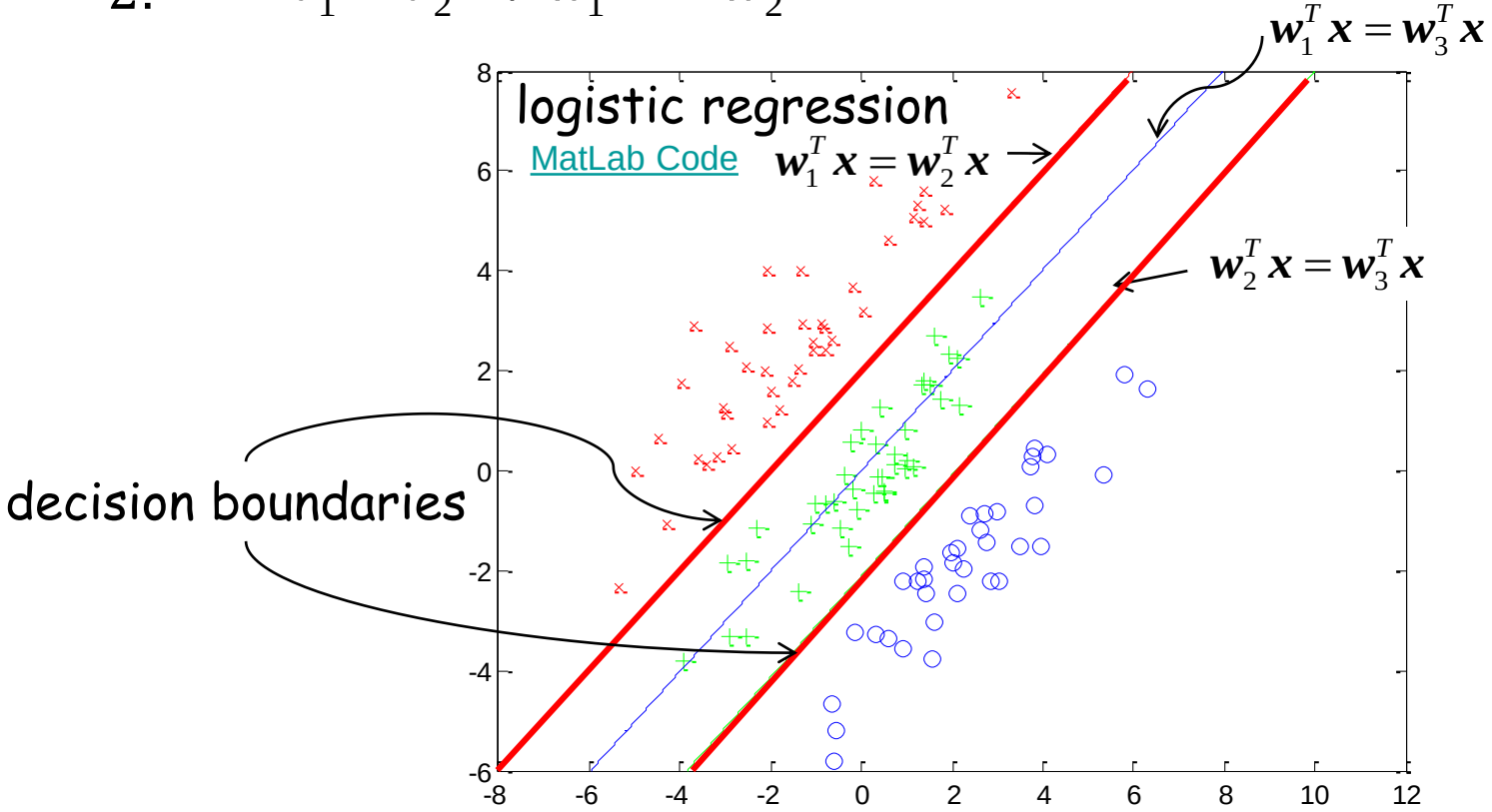


Logistic Regression for Multiclass Classification

- For logistic regression we work with posterior class probabilities

$$p(C_k|\boldsymbol{x}) = \exp(a_k) \Big/ \sum_j \exp(a_j)$$

- At the decision boundaries (separating 2 classes), the corresponding posteriors are equal, e.g between classes 1 and 2: $a_1 = a_2 \Rightarrow \boldsymbol{w}_1^T \boldsymbol{x} = \boldsymbol{w}_2^T \boldsymbol{x}$



Generative Vs. Discriminative Classifiers

Generative Versus Discriminative Classifiers

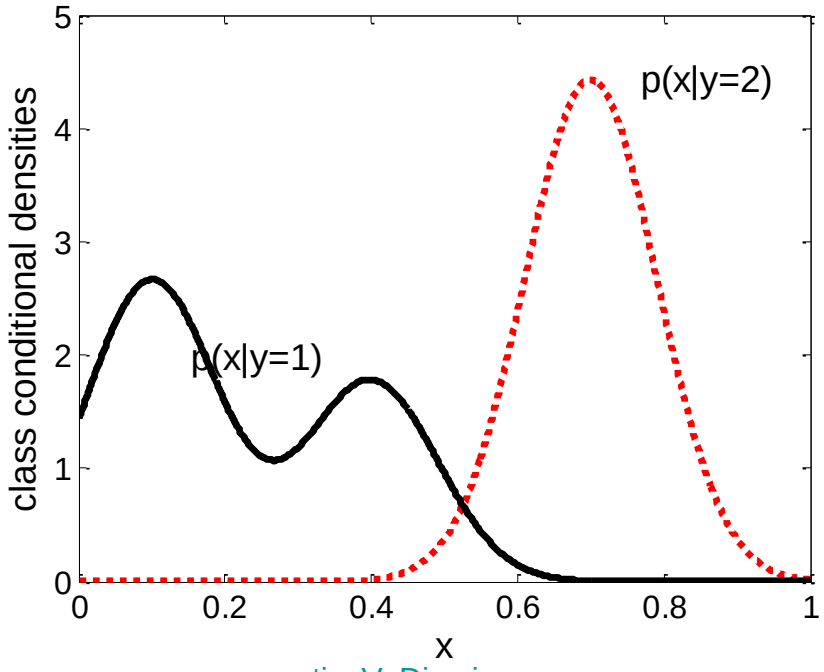
- *The assumptions made by GDA are much stronger than the assumptions made by logistic regression.*
  - When fitting a discriminative model, we usually maximize the conditional log likelihood $\sum_{i=1}^{N} logp(y_i|\boldsymbol{x}_i, \boldsymbol{\theta})$, whereas
  - when fitting a generative model, we usually maximize the joint log likelihood, $\sum_{i=1}^{N} logp(y_i, \boldsymbol{x}_i|\boldsymbol{\theta})$.
  - In general they give different results.
- When the Gaussian assumptions made by GDA are correct, the model will need less training data than logistic regression, but if the Gaussian assumptions are incorrect, logistic regression will do better.*
- This is because discriminative models do not need to model the distribution of the features. This is illustrated next.

❖ Ng, A. Y. and M. I. Jordan (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS- 14*.

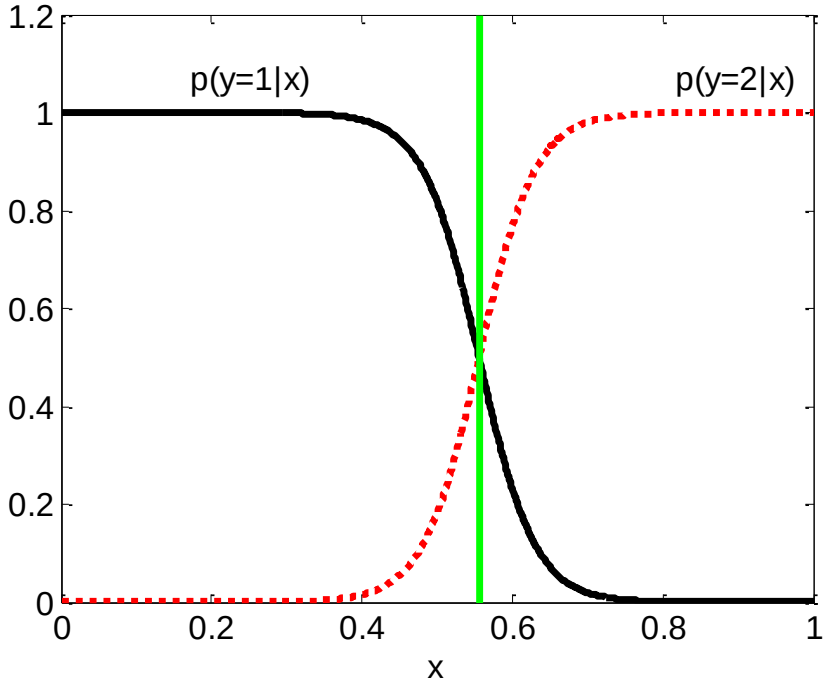
Generative Versus Discriminative Classifiers

- The class conditional densities are complex (here $p(x|y = 1)$ is a multimodal distribution), but the class posterior, $p(y = c|x)$, is a simple sigmoidal function, centered on the threshold value of 0.55.

- In general, *discriminative methods will be more accurate.*



generativeVsDiscrim
from Kevin Murphys' PMTK



Generative Versus Discriminative Classifiers

- Easy to fit generative classifiers: e.g. in naive Bayes model and an LDA model by simple counting/averaging vs complex optimization in logistic regression.
- Adding new classes in a generative classifier is not a problem: we estimate the parameters of each class conditional density independently (not the case in discriminative models).
- In generative classifiers, we can deal easily with missing features: However, in a discriminative classifier, there is no principled solution to this problem, since the model is conditioned on $\boldsymbol{x}$.

$$p\ \ y=c\,|\,\boldsymbol{x}_{2:D},\boldsymbol{\theta}\ \propto p\ \ y=c\,|\,\boldsymbol{\theta}\ \ p\ \ \boldsymbol{x}_{2:D}\,|\,y=c,\boldsymbol{\theta}\ = p\ \ y=c\,|\,\boldsymbol{\theta}\ \sum_{x_1} p\ \ x_1,\boldsymbol{x}_{2:D}\,|\,y=c,\boldsymbol{\theta}$$

- Salojarvi, J., K. Puolamaki, and S. Klaski (2005). On discriminative joint density modeling. In *Proc. European Conf. on Machine Learning*
- Marlin, B. (2008). *Missing Data Problems in Machine Learning*. Ph.D. thesis, U. Toronto
- Lasserre, J., C. Bishop, and T. Minka (2006). Principled hybrids of generative and discriminative models. In *CVPR*.
- Liang, F., S. Mukherjee, and M. West (2007). Understanding the use of unlabelled data in predictive modelling. *Statistical Science 22*, 189– 205.
- Little., R. J. and D. B. Rubin (1987). *Statistical Analysis with Missing Data*. New York: Wiley and Son.

Generative Versus Discriminative Classifiers

$$p\ \ y=c\,|\,\boldsymbol{x}_{2:D},\boldsymbol{\theta}\ \propto p\ \ y=c\,|\,\boldsymbol{\theta}\ \ p\ \ \boldsymbol{x}_{2:D}\,|\,y=c,\boldsymbol{\theta}\ = p\ \ y=c\,|\,\boldsymbol{\theta}\ \sum_{x_1} p\ \ x_1,\boldsymbol{x}_{2:D}\,|\,y=c,\boldsymbol{\theta}$$

- This is further simplified if we employ the naïve Bayes assumption

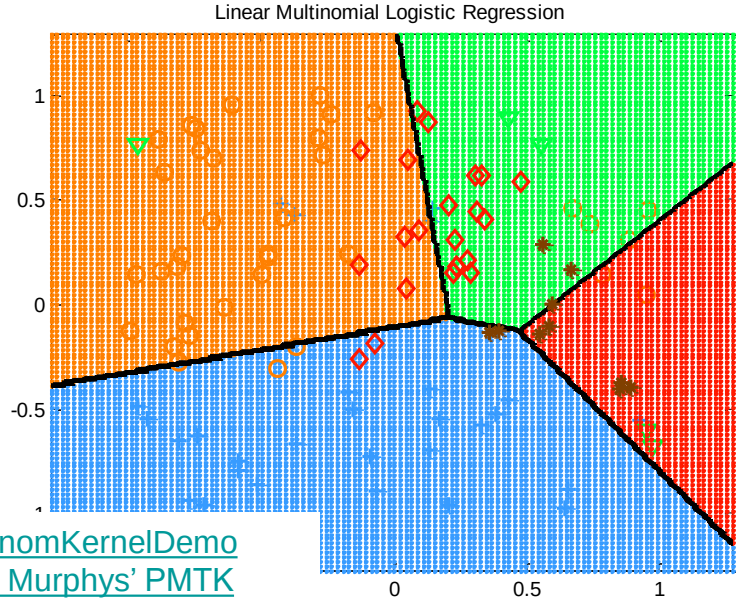
$$\sum_{x_1} p\ \ x_1,\boldsymbol{x}_{2:D}\,|\,y=c,\boldsymbol{\theta}\ =\sum_{x_1} p\ \ x_1\,|\,y=c,\boldsymbol{\theta}_{1c}\ \prod_{j=2}^{D} p\ \ x_j\,|\,y=c,\boldsymbol{\theta}_{jc}\ =\prod_{j=2}^{D} p\ \ x_j\,|\,y=c,\boldsymbol{\theta}_{jc}$$

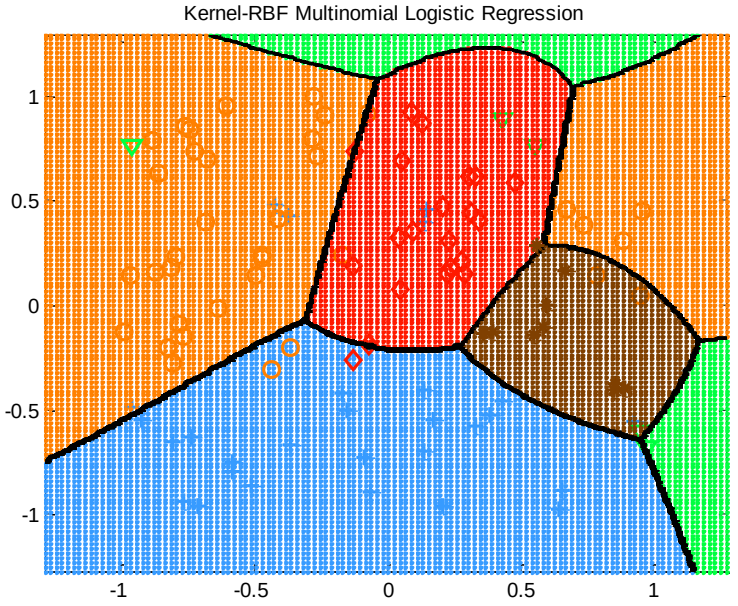
- In discriminative analysis, we can analytically marginalize missing values, e.g. $p(\boldsymbol{x}_{2:D}|y=c,\boldsymbol{\theta})=\mathcal{N}\left(\boldsymbol{x}_{2:D}|\mu_{c,2:D},\boldsymbol{\Sigma}_{c,2:D,2:D}\right)$

- Using unlabeled data in semi-supervised learning is easy with generative models: A much harder task in discriminative models.

Generative Versus Discriminative Classifiers

- Generative models can generate inputs given the outputs: *we can infer probable inputs given the output* $p(\boldsymbol{x}|y)$. This is not possible with a discriminative model. A generative model defines $p(\boldsymbol{x}, y)$.
- *Discriminative methods allow preprocessing of the input:* replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$. This cannot be done in generative models.
- *Discriminative Models – Better calibrated in terms of probability*: Independence assumptions in generative models (e.g. naive Bayes) lead to extreme posterior class probabilities (near 0 or 1).





logregMultinomKernelDemo
from Kevin Murphys' PMTK

Fisher's Linear Discriminant

Fisher's Linear Discriminant

- We now view a linear classification model in terms of dimensionality reduction. Consider two classes, and suppose we take the D −dimensional input $\boldsymbol{x}$ and project it to 1D:

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

- If we place a threshold on y and classify $y \geq -w_0$ as class C_1, and otherwise class C_2, then we obtain our standard linear classifier.
- The projection onto 1D leads to loss of information:
  - classes that are well separated in the original D-dimensional input space may become strongly overlapping in 1D.
- By adjusting $\boldsymbol{w}$, can we select a projection that maximizes the class separation?

Fisher's Linear Discriminant

- Consider a two-class problem in which there are N_1 points of class C_1 and N_2 points of class C_2, so that the mean vectors of the two classes are given by

Class means: $$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in C_1}\boldsymbol{x}_n, \quad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in C_2}\boldsymbol{x}_n$$

- A measure of the separation of the two classes when projected onto $\boldsymbol{w}$ is the separation of the projected means:

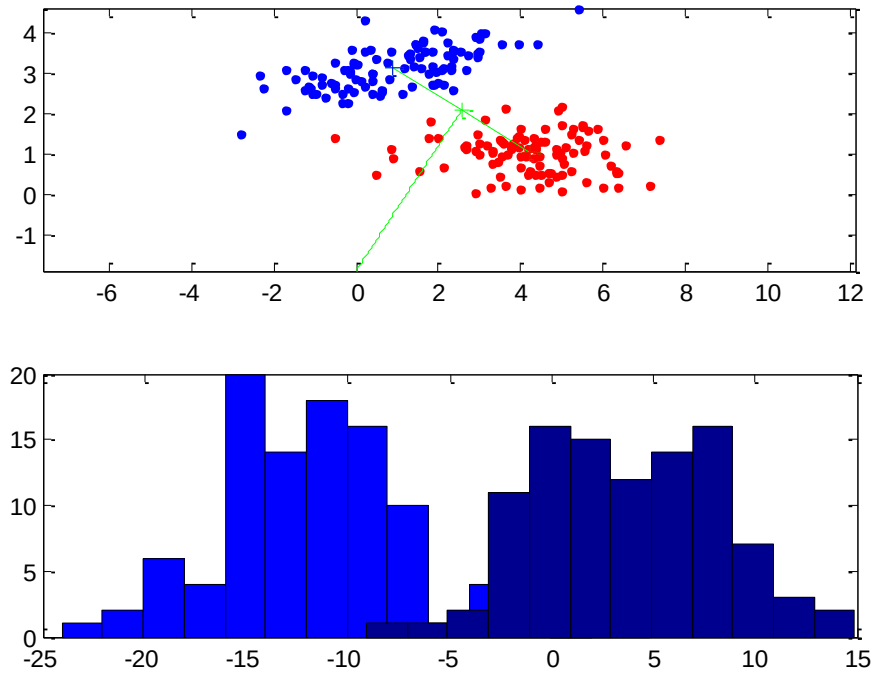
$$m_2 - m_1 = \boldsymbol{w}^T\left(\boldsymbol{m}_2 - \boldsymbol{m}_1\right), \ \ where \ \ m_i = \boldsymbol{w}^T\boldsymbol{m}_i, i = 1,2, \sum_i w_i^2 = 1$$

- The separation $\boldsymbol{w}^T\left(\boldsymbol{m}_2 - \boldsymbol{m}_1\right)$ can be made arbitrarily large by increasing $|\boldsymbol{w}|$, so we constrain $\boldsymbol{w}$ as shown.

- Using a Lagrange multiplier to perform the constrained maximization we find that $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$.

Fisher's Linear Discriminant

- We show here 2 classes that are well separated in the (x_1, x_2) space but overlap when projected onto $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$.

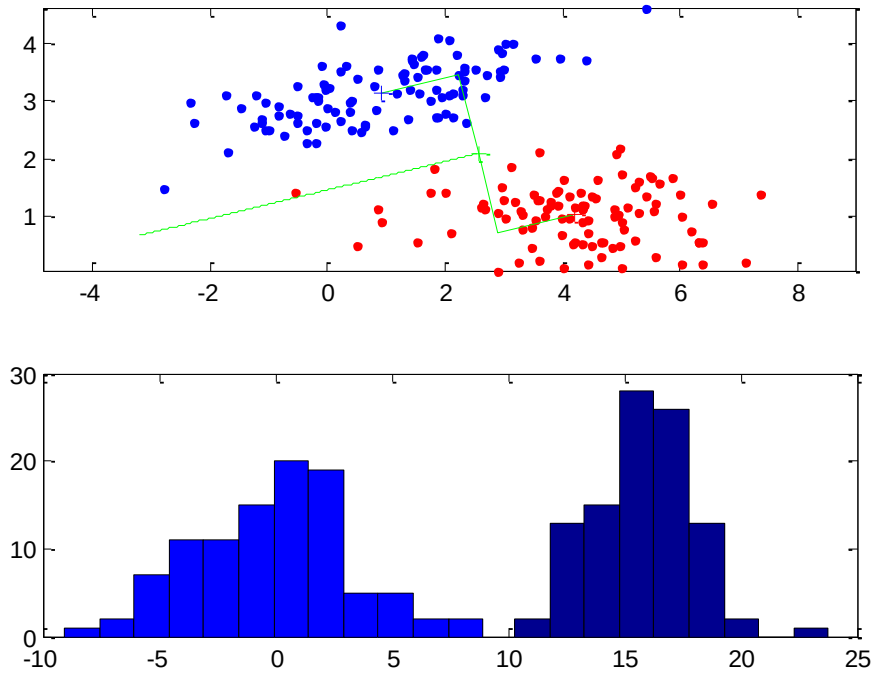


MatLab Code

- This is due to the *strongly non-diagonal covariances of the class distributions.*

Fisher's Linear Discriminant

- We showed, in the last slide, two well separated classes in (x_1, x_2) space that overlap when projected onto $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$.

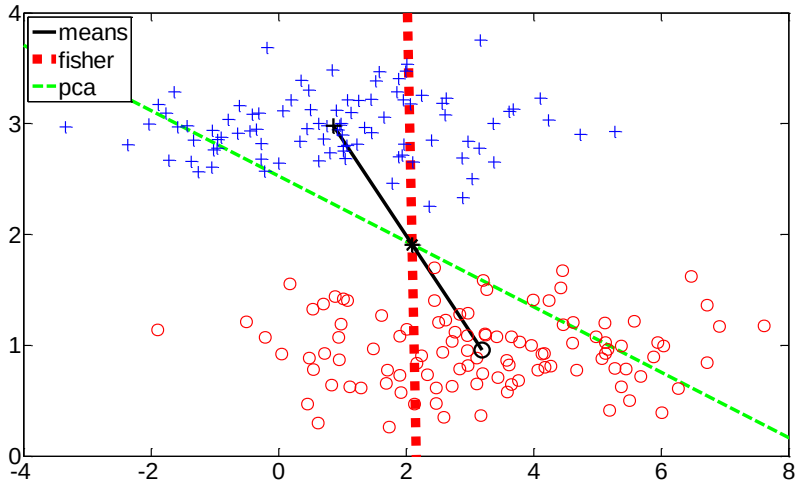


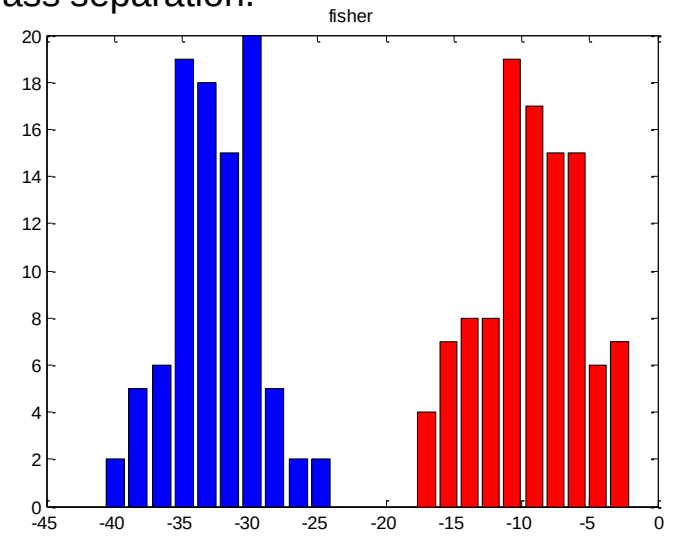
MatLab Code

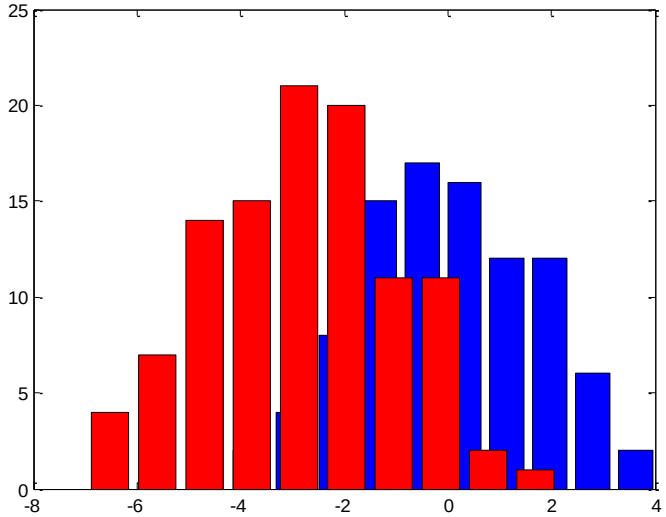
- Fisher proposed instead to maximize a function that gives a large separation between the projected class means while also giving a small variance within each class, thereby *minimizing the class overlap.*

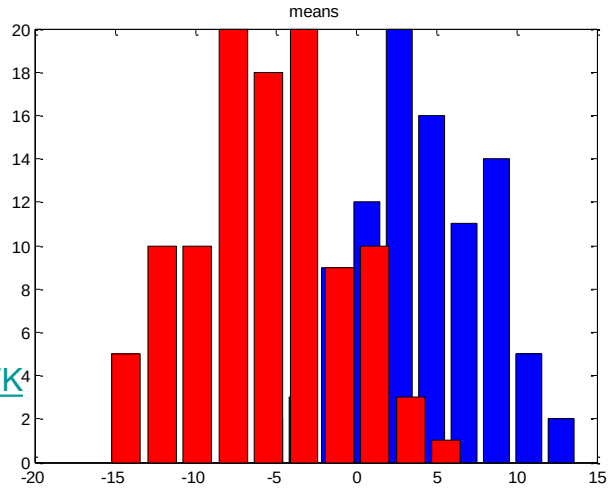
Fisher's Linear Discriminant

- (a) Comparison of class classification using 1st PCA vector, Fishers' linear discriminant vector and class-conditional means. (b) Projection of points onto Fisher's vector shows good class separation. (c) Projection of points onto PCA vector shows poor class separation.









fisherLDAdemo
Kevin Murphys' PMTK

Fisher's Linear Discriminant

- The within-class variance of the transformed data from class C_k is given by

$$s_k^2 = \sum_{n \in C_k} \left(y_n - m_k \right)^2, \ \ where \ \ m_k = \boldsymbol{w}^T \boldsymbol{m}_k, k = 1, 2, and \ \ y_n = \boldsymbol{w}^T \boldsymbol{x}_n$$

- We define the total within-class variance as $s_1^2 + s_2^2$
- The Fisher criterion is then defined as the ratio of the between-class variance to the within-class variance for the transformed data as:

$$J(\boldsymbol{w}) = \frac{\left(m_1 - m_2 \right)^2}{s_1^2 + s_2^2} = \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}}$$

where the between-class and within class covariances for the initial data are defined as:

$$\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T, \ \boldsymbol{S}_W = \sum_{n \in C_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n \in C_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

Fisher's Linear Discriminant

- Differentiating wrt $\boldsymbol{w}$ gives:

$$\frac{d}{d\boldsymbol{w}} J(\boldsymbol{w}) = \frac{d}{d\boldsymbol{w}} \frac{\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}} = 0 \Rightarrow \left(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}\right) \boldsymbol{S}_W \boldsymbol{w} = \left(\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}\right) \boldsymbol{S}_B \boldsymbol{w}$$

- From $\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$, we see that $\boldsymbol{S}_B \boldsymbol{w}$ is in the direction of $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$. Since we only care about the direction of $\boldsymbol{w}$, we drop the scalars $\left(\boldsymbol{w}^T \boldsymbol{S}_B \boldsymbol{w}\right), \left(\boldsymbol{w}^T \boldsymbol{S}_W \boldsymbol{w}\right)$ and multiplying by $\boldsymbol{S}_W^{-1}$ gives:

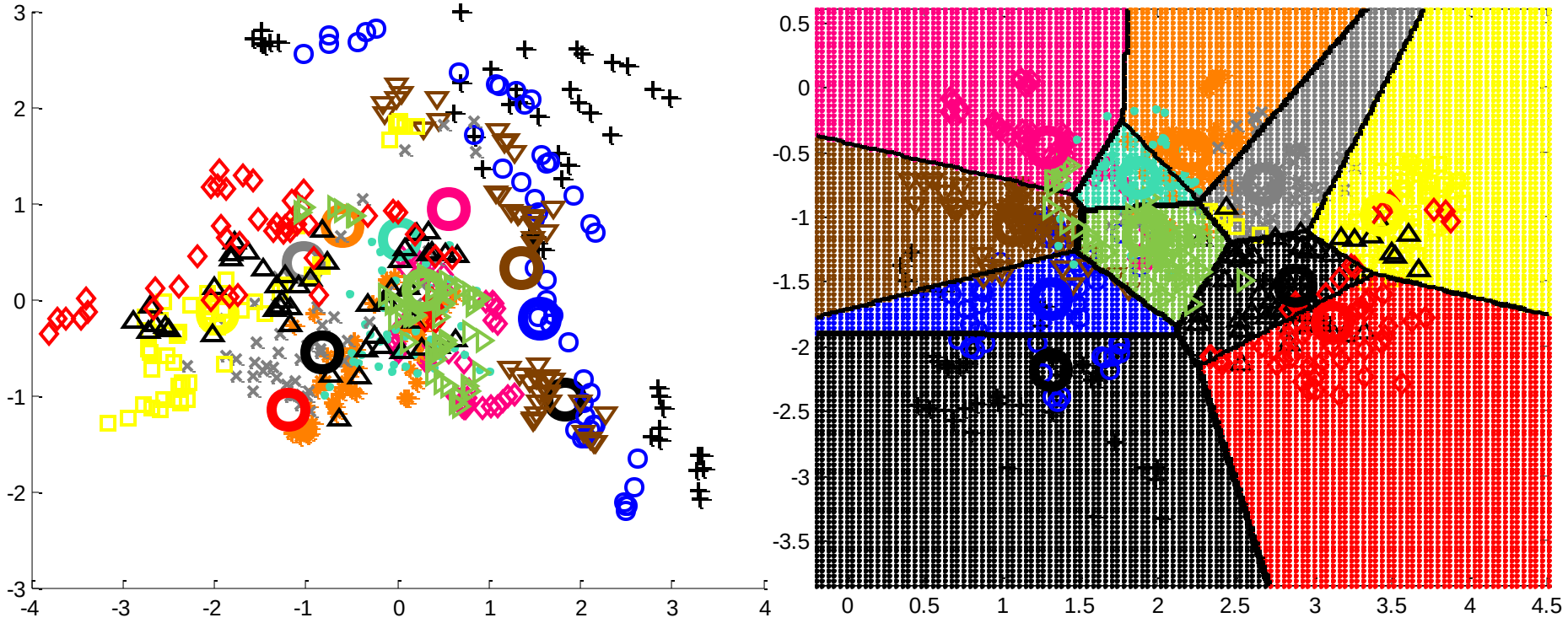
$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

$$\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T, \boldsymbol{S}_W = \sum_{n \in C_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n \in C_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

- Note that if the within-class covariance $\boldsymbol{S}_W$ is isotropic, $\boldsymbol{S}_W \propto \boldsymbol{I}$, then $\boldsymbol{w} \propto (\boldsymbol{m}_2 - \boldsymbol{m}_1)$ as discussed earlier.

Fisher's Linear Discriminant

- (a) PCA projection of vowel data (10-D) to 2d. (b) FLDA projection of vowel data to 2d. There is better class separation in the FLDA case.



fisherDiscrimVowelDemo
from Kevin Murphys' PMTK

- Hastie, T., R. Tibshirani, and J. Friedman (2009). *The Elements of Statistical Learning*. Springer. 2nd edition.

Fisher's Linear Discriminant

$$\boldsymbol{w} \propto \boldsymbol{S}_W^{-1}(\boldsymbol{m}_2 - \boldsymbol{m}_1)$$

- The Fisher's linear discriminant is a choice of direction for projection of the data to 1D.
- The projected data can be used to construct a discriminant by choosing a threshold y_0 so that we classify a new point belonging to C_1 if $y(\boldsymbol{x}) \geq y_0$ and to C_2 otherwise.
- To choose the threshold, *we model the class-conditional densities $p(y|C_k)$ using Gaussians and then use MLE to estimate their parameters.*
- Using the Gaussian approximations to the projected classes, Decision theory (minimizing the misclassification rate) gives an expression for the optimal threshold.
- Justification for the Gaussian assumption comes from the CLT noting that $y = \boldsymbol{w}^T\boldsymbol{x}$ is sum of a set of random variables.

Fisher's Discriminant vs. Least Squares

- The least-squares approach to determining a linear discriminant was based on making the model predictions as close as possible to the target values (using the 1-of-K coding).
- By contrast, the Fisher criterion was derived by requiring maximum class separation in the output space.
- For the two-class problem, the Fisher criterion can be obtained as a special case of least squares.
- Unfortunately we need to introduce a different target coding scheme (from the 1-of-K coding) for the least-squares solution for $\boldsymbol{w}$ to become equivalent to the Fisher solution.

Fisher's Discriminant vs. Least Squares

- We take the targets for class C_1 to be N/N_1, where N_1 is the number of patterns in class C_1, and N is the total number of patterns. *This approximates the reciprocal of the prior probability for class C_1.*
- *For class C_2, we take the targets to be $-N/N_2$,* where N_2 is the number of patterns in class C_2. With this choice:

$$\sum_{n=1}^{N} t_n = N_1 \frac{N}{N_1} + N_2\left(-\frac{N}{N_2}\right) = 0$$

- The sum-of-squares error function is written as

$$E = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - t_n\right)^2$$

- Taking the derivatives wrt w_0 and $\boldsymbol{w}$ gives:

$$\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - t_n\right) = 0 \Rightarrow N\boldsymbol{w}^T\boldsymbol{m} + Nw_0 - \sum_{n=1}^{N} t_n = 0 \Rightarrow w_0 = -\boldsymbol{w}^T\boldsymbol{m}, \boldsymbol{m} = \frac{\sum_{n=1}^{N}\boldsymbol{x}_n}{N} = \frac{N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2}{N}$$

$$\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - t_n\right)\boldsymbol{x}_n = 0 \Rightarrow \left(\boldsymbol{S}_w + \frac{N_1N_2}{N}\boldsymbol{S}_B\right)\boldsymbol{w} = N\left(\boldsymbol{m}_1 - \boldsymbol{m}_2\right)$$ (see proof on next slide)

Fisher's Discriminant vs. Least Squares

$$\sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{x}_n + w_0 - t_n\right)\boldsymbol{x}_n^T = 0, \text{ where } w_0 = -\boldsymbol{w}^T\boldsymbol{m}, \ \boldsymbol{m} = \frac{N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2}{N} \text{ and}$$

$$\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T, \ \boldsymbol{S}_W = \sum_{n\in C_1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \sum_{n\in C_2}(\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T$$

- The first Equ. can be simplified as:

$$\boldsymbol{w}^T\sum_{n\in C_1}\boldsymbol{x}_n\boldsymbol{x}_n^T + \boldsymbol{w}^T\sum_{n\in C_2}\boldsymbol{x}_n\boldsymbol{x}_n^T - \boldsymbol{w}^T\boldsymbol{m}\left(N\boldsymbol{m}^T\right) - \sum_{n=1}^{N}t_n\boldsymbol{x}_n^T = 0$$

- Note from our choice of target values:

$$\sum_{n=1}^{N}t_n\boldsymbol{x}_n^T = \frac{N}{N_1}\sum_{n\in C_1}\boldsymbol{x}_n^T - \frac{N}{N_2}\sum_{n\in C_2}\boldsymbol{x}_n^T = N\left(\boldsymbol{m}_1^T - \boldsymbol{m}_2^T\right)$$

and also

$$\sum_{n\in C_1}\boldsymbol{x}_n\boldsymbol{x}_n^T = \sum_{n\in C_1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + \boldsymbol{m}_1\sum_{n\in C_1}\boldsymbol{x}_n^T + \sum_{n\in C_1}\boldsymbol{x}_n\boldsymbol{m}_1^T - N_1\boldsymbol{m}_1\boldsymbol{m}_1^T =$$

$$= \sum_{n\in C_1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + N_1\boldsymbol{m}_1\boldsymbol{m}_1^T$$

Fisher's Discriminant vs. Least Squares

$$\boldsymbol{w}^T \sum_{n\in C_1} \boldsymbol{x}_n \boldsymbol{x}_n^T + \boldsymbol{w}^T \sum_{n\in C_2} \boldsymbol{x}_n \boldsymbol{x}_n^T - \boldsymbol{w}^T \boldsymbol{m}\left(N\boldsymbol{m}^T\right) - \sum_{n=1}^{N} t_n \boldsymbol{x}_n^T = 0$$

- We now simplify the above equation as:

$$\boldsymbol{w}^T \left\{ \sum_{n\in C_1} (\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^T + N_1 \boldsymbol{m}_1 \boldsymbol{m}_1^T + \sum_{n\in C_2} (\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^T + N_2 \boldsymbol{m}_2 \boldsymbol{m}_2^T - \boldsymbol{m}\left(N\boldsymbol{m}^T\right) \right\}$$

$$-N\left(\boldsymbol{m}_1^T - \boldsymbol{m}_2^T\right) = 0$$

or using the definition of $\boldsymbol{S}_W$

$$\boldsymbol{w}^T \left\{ \boldsymbol{S}_W + N_1 \boldsymbol{m}_1 \boldsymbol{m}_1^T + N_2 \boldsymbol{m}_2 \boldsymbol{m}_2^T - N\boldsymbol{m}\boldsymbol{m}^T \right\} = N\left(\boldsymbol{m}_1^T - \boldsymbol{m}_2^T\right) \Rightarrow$$

$$\boldsymbol{w}^T \left\{ \boldsymbol{S}_W + \underbrace{N_1 \boldsymbol{m}_1 \boldsymbol{m}_1^T + N_2 \boldsymbol{m}_2 \boldsymbol{m}_2^T - N\left(\frac{N_1}{N}\boldsymbol{m}_1 + \frac{N_2}{N}\boldsymbol{m}_2\right)\left(\frac{N_1}{N}\boldsymbol{m}_1^T + \frac{N_2}{N}\boldsymbol{m}_2^T\right)}_{\frac{N_1 N_2}{N}(\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T = \frac{N_1 N_2}{N}\boldsymbol{S}_B} \right\} = N\left(\boldsymbol{m}_1^T - \boldsymbol{m}_2^T\right) \Rightarrow$$

$$\left(\boldsymbol{S}_w + \frac{N_1 N_2}{N}\boldsymbol{S}_B\right)\boldsymbol{w} = N\left(\boldsymbol{m}_1 - \boldsymbol{m}_2\right)$$

Fisher's discriminant vs. Least Squares

$$\left(\boldsymbol{S}_w + \frac{N_1 N_2}{N} \boldsymbol{S}_B \right) \boldsymbol{w} = N \left(\boldsymbol{m}_1 - \boldsymbol{m}_2 \right)$$

- Using $\boldsymbol{S}_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^T$, we note that $\boldsymbol{S}_B \boldsymbol{w}$ is in the direction of $(\boldsymbol{m}_2 - \boldsymbol{m}_1)$. Thus we can write

$$\boldsymbol{w} \propto \boldsymbol{S}_w^{-1} \left(\boldsymbol{m}_2 - \boldsymbol{m}_1 \right)$$

- This is the same result obtained with the Fisher criterion!

- We also found the bias $w_0 = -\boldsymbol{w}^T \boldsymbol{m}$. This tells us that a new vector $\boldsymbol{x}$ should be classified as belonging to C_1 if

$$y(\boldsymbol{x}) = w_0 + \boldsymbol{w}^T \boldsymbol{x} = -\boldsymbol{w}^T \boldsymbol{m} + \boldsymbol{w}^T \boldsymbol{x} = \boldsymbol{w}^T \left(\boldsymbol{x} - \boldsymbol{m} \right) > 0$$

and class C_2 otherwise.

Fisher's Discriminant for Multiple Classes

- We now consider the generalization of the Fisher discriminant to $K > 2$ classes.
- Assume the dimensionality of the input space is $D > K$.
- We introduce $D' > 1$ linear 'features' $y_k = \boldsymbol{w}_k^T \boldsymbol{x}, k = 1,...,D'$. We group these feature values in a vector $\boldsymbol{y}$.
- Similarly, the weight vectors $\{\boldsymbol{w}_k\}$ are taken as the columns of a matrix $\boldsymbol{W}$, so that

$$\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}$$

- Note that here we are not including any bias parameters in the definition of $\boldsymbol{y}$.
- The within-class covariance matrix for K classes is:

$$\boldsymbol{S}_W = \sum_{k=1}^{K} \boldsymbol{S}_k, \boldsymbol{S}_k = \sum_{n \in C_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T, \boldsymbol{m}_k = \frac{1}{N_k} \sum_{n \in C_k} \boldsymbol{x}_n$$

Fisher's Discriminant for Multiple Classes

- We need to also find the between-class covariance matrix. We work here in the $\boldsymbol{x}$ –space. Start with the total covariance:

$$\boldsymbol{S}_T = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^T, \;\; \boldsymbol{m} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n = \frac{1}{N}\sum_{k=1}^{K} N_k \boldsymbol{m}_k, \;\; \sum_{k=1}^{K} N_k = N$$

- The total covariance matrix can be decomposed into the sum of the within-class covariance matrix plus an additional matrix $\boldsymbol{S}_B$, which we identify as a measure of the between-class covariance:

$$\boldsymbol{S}_T = \boldsymbol{S}_W + \boldsymbol{S}_B, \;\; \boldsymbol{S}_W = \sum_{k=1}^{K} \boldsymbol{S}_k, \; \boldsymbol{S}_k = \sum_{n \in C_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T, \; \boldsymbol{m}_k = \frac{1}{N_k}\sum_{n \in C_k} \boldsymbol{x}_n$$

- From the above two Eqs we can show that:

$$\boldsymbol{S}_B = \sum_{k}\sum_{n \in C_k} (\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^T - \sum_{k}\sum_{n \in C_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^T = \sum_{k=1}^{K} N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T$$

Fisher's Discriminant for Multiple Classes

- We can define similar covariance matrices in the D' dimensional $\boldsymbol{y}$ −space:

$$\boldsymbol{s}_W = \sum_{k=1}^{K} \sum_{n \in C_k} (\boldsymbol{y}_n - \boldsymbol{\mu}_k)(\boldsymbol{y}_n - \boldsymbol{\mu}_k)^T, \;\; \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in C_k} \boldsymbol{y}_n$$

$$\boldsymbol{s}_B = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T, \;\; \boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k$$

- We wish to maximize the between-class covariance while minimizing the within-class covariance. There are many choices of (scalar) criteria:

$$J(\boldsymbol{W}) = Tr\left\{\boldsymbol{s}_W^{-1}\boldsymbol{s}_B\right\} = Tr\left\{\left(\boldsymbol{W}^T\boldsymbol{S}_W\boldsymbol{W}\right)^{-1}\left(\boldsymbol{W}^T\boldsymbol{S}_B\boldsymbol{W}\right)\right\}$$

- Maximization is straightforward. The weight values are determined as $\boldsymbol{W} = \boldsymbol{S}_W^{-1/2}\boldsymbol{U}$ where U are the D' leading eigenvectors of $\boldsymbol{S}_W^{-1/2}\boldsymbol{S}_B\boldsymbol{S}_W^{-1/2}$ (D' = number of linear features).

Fisher's Discriminant for Multiple Classes

- Note $\boldsymbol{S}_B = \sum_{k=1}^{K} N_k (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^T$ is composed of the sum of K matrices, each of which is an outer product of two vectors and therefore of rank 1.

- However, only $(K - 1)$ of these matrices are independent since $\boldsymbol{m} = \sum_{k=1}^{K} N_k \boldsymbol{m}_k \Big/ N$. Thus, $\boldsymbol{S}_B$ has rank at most $(K - 1)$ i.e. has at most $(K - 1)$ nonzero eigenvalues.

- Projecting onto the $(K - 1)$ –dim subspace spanned by the eigenvectors of $\boldsymbol{S}_B$ does not change the value of $J(\boldsymbol{W})$.

  - *Severe Limitation of FLDA: We can find at most $L = K - 1$ linear features regardless of the dimensionality of $\boldsymbol{x}$.*

- Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition* (Second ed.). Academic Press.

Probabilistic Interpretation of FLDA

- A probabilistic interpretation of FLDA was proposed.[1,2]
- In the ***heteroscedastic LDA*** *(HLDA),* let $\boldsymbol{W}$ be a $D \times D$ invertible matrix and let $\boldsymbol{z}_i = \boldsymbol{W}\boldsymbol{x}_i$ a transformed version of the data.
- We *fit full covariance Gaussians to the transformed data, one per class*, but with the constraint that *only the first L components will be class-specific.*
- *T*he remaining $H = D - L$ components are shared across classes, and will thus not be discriminative.

$$p(\boldsymbol{z}_i \mid \boldsymbol{\theta}, y_i = c) = \mathcal{N}(z_i \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \boldsymbol{\mu}_c = \begin{pmatrix} \boldsymbol{m}_c \\ \boldsymbol{m}_0 \end{pmatrix}, \boldsymbol{\Sigma}_c = \begin{pmatrix} \boldsymbol{S}_c & 0 \\ 0 & \boldsymbol{S}_0 \end{pmatrix}$$

1. Kumar, N. and A. Andreo (1998). Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech Communication 26*, 283–297.
2. Zhou, H., D. Karakos, S. Khudanpur, A. Andreou, and C. Priebe (2009). On Projections of Gaussian Distributions using Maximum Likelihood Criteria. In *Proc. of the Workshop on Information Theory and its Applications*.
3. Gales, M. (2002). Maximum likelihood multiple subspace projections for hidden Markov models. *IEEE. Trans. on Speech and Audio Processing 10*(2), 37–47.
4. Gales, M. J. F. (1999). Semi-tied covariance matrices for hidden Markov models. *IEEE Trans. on Speech and Audio Processing 7* (3), 272–281.

Probabilistic Interpretation of FLDA

$$p\ \mathbf{z}_i \mid \theta, y_i = c\ = \mathcal{N}\ \mathbf{z}_i \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\ , \boldsymbol{\mu}_c = \begin{pmatrix} \boldsymbol{m}_c \\ \boldsymbol{m}_0 \end{pmatrix}, \boldsymbol{\Sigma}_c = \begin{pmatrix} \boldsymbol{S}_c & 0 \\ 0 & \boldsymbol{S}_0 \end{pmatrix}$$

- Here $\boldsymbol{m}_0$ is the shared H dimensional mean and $\boldsymbol{S}_0$ is the shared $H \times H$ covariace.

- The pdf of the original (untransformed) data is then:

$$p\ \mathbf{x}_i \mid y_i = c, \boldsymbol{W}, \boldsymbol{\theta}\ = |\boldsymbol{W}|\, \mathcal{N}\ \boldsymbol{W}\mathbf{x}_i \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\ = |\boldsymbol{W}|\, \mathcal{N}\ \boldsymbol{W}_L \mathbf{x}_i \mid \boldsymbol{m}_c, \boldsymbol{S}_c\ \mathcal{N}\ \boldsymbol{W}_H \mathbf{x}_i \mid \boldsymbol{m}_0, \boldsymbol{S}_0\ , \boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_L \\ \boldsymbol{W}_H \end{pmatrix}$$

- For fixed $\boldsymbol{W}$, it is easy to derive the MLE for $\boldsymbol{\theta}$. One can then optimize $\boldsymbol{W}$ using gradient methods.

Probabilistic Interpretation of FLDA

$$p\ \ z_i \mid \theta, y_i = c\ \ = \mathcal{N}\ \ z_i \mid \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\ \ , \boldsymbol{\mu}_c = \begin{pmatrix} \boldsymbol{m}_c \\ \boldsymbol{m}_0 \end{pmatrix}, \boldsymbol{\Sigma}_c = \begin{pmatrix} \boldsymbol{S}_c & 0 \\ 0 & \boldsymbol{S}_0 \end{pmatrix}$$

- When $\boldsymbol{\Sigma}_c$ are diagonal, there is a closed-form for $\boldsymbol{W}$.[2]
- In the special case the $\boldsymbol{\Sigma}_c$ are all equal, we recover classical LDA.[4]
- HLDA outperforms LDA if the class covariances are not equal within the discriminative subspace (i.e., if the assumption that $\boldsymbol{\Sigma}_c$ is independent of *c* is a poor assumption). This was demonstrated on speech recognition.[3]
- One can allow each class to use its own projection matrix ***(multiple LDA)***.[1]

1. Gales, M. (2002). Maximum likelihood multiple subspace projections for hidden Markov models. *IEEE. Trans. on Speech and Audio Processing 10*(2), 37–47
2. Gales, M. J. F. (1999). Semi-tied covariance matrices for hidden Markov models. *IEEE Trans. on Speech and Audio Processing 7* (3), 272–281.
3. Kumar, N. and A. Andreo (1998). Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech Communication 26*, 283–297.
4. Zhou, H., D. Karakos, S. Khudanpur, A. Andreou, and C. Priebe (2009). On Projections of Gaussian Distributions using Maximum Likelihood Criteria. In *Proc. of the Workshop on Information Theory and its Applications*.

Probabilistic Generative Models

Probabilistic Generative Models

- Models with linear decision boundaries arise from simple assumptions about the probabilistic distribution of the data.

- We here adopt **a generative approach:** we model the class-conditional densities $p(\boldsymbol{x}|C_k)$, as well as the class priors $p(C_k)$, and then use these to compute posterior probabilities $p(C_k|\boldsymbol{x})$ through Bayes' theorem.

- Let us consider two classes:

$$p(C_1 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_1)\,p(C_1)}{p(\boldsymbol{x} \mid C_1)\,p(C_1) + p(\boldsymbol{x} \mid C_2)\,p(C_2)} = \frac{1}{1+e^{-a}} = \sigma(a)$$

where:

$$a = \ln \frac{p(\boldsymbol{x} \mid C_1)\,p(C_1)}{p(\boldsymbol{x} \mid C_2)\,p(C_2)} \quad \text{and} \quad \sigma(a) = \frac{1}{1+e^{-a}}$$

Logistic sigmoid

Logistic Sigmoid (S-shaped)

- We can approximate the logistic sigmoid function with the scaled inverse probit function.
- The derivatives of the two curves are the same for $a = 0$ (selection of $\lambda^2 = \frac{\pi}{8}$).

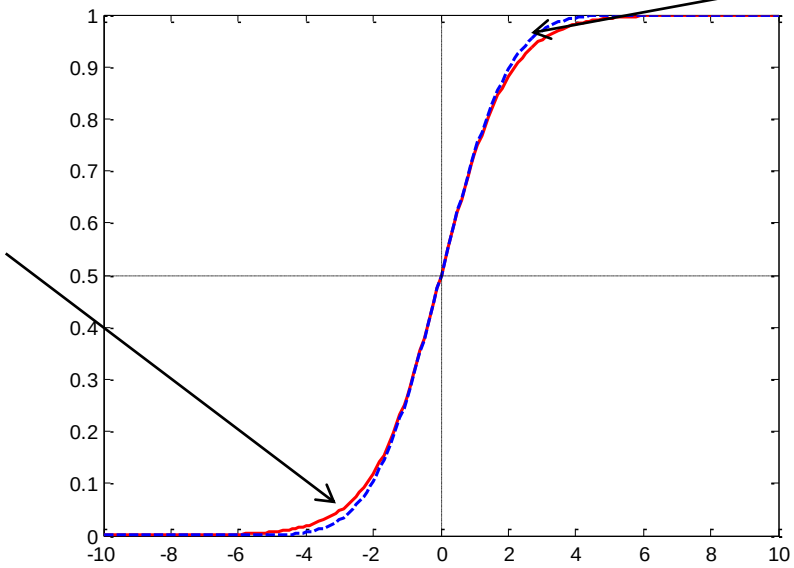
$$\Phi(\lambda a) = \int_{-\infty}^{\lambda a} \mathcal{N}(\theta \mid 0,1)\, d\theta$$

-- dashed line

Scaled inverse Probit function

Logistic sigmoid

$$\sigma(a) = \frac{1}{1+e^{-a}}$$



MatLab Code

- The logistic sigmoid function satisfies: $\sigma(-a) = 1 - \sigma(a)$
- The *inverse of $\sigma(a)$ is the logit function*: $a = \ln \frac{\sigma}{1-\sigma} = \ln \frac{p(C_1 \mid \boldsymbol{x})}{p(C_2 \mid \boldsymbol{x})}$

Probabilistic Generative Models

- The appearance of the logistic sigmoid is rather arbitrary up to this point.

$$p(C_1 \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_1)p(C_1) + p(\boldsymbol{x} \mid C_2)p(C_2)} = \frac{1}{1+e^{-a}} = \sigma(a)$$

- This will be useful provided $a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_2)p(C_2)}$ takes a simple functional form.

- We will consider problems with $a(\boldsymbol{x})$ being a linear function of $\boldsymbol{x}$.

- In this case, the posterior probability is governed by a generalized linear model.

Probabilistic Generative Models $K > 2$

- For the case of $K > 2$ *classes, we have*

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)\, p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)\, p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}}, \; a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- This is known as the normalized exponential and can be regarded as a multiclass generalization of the logistic sigmoid.
- Here we define:

$$a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- The normalized exponential is known as the softmax function.
- It can be seen as a smoothed version of the max function because, if $a_k \geq a_j$ for all $j \neq k$, then $p(C_k|\boldsymbol{x}) \cong 1$, and $p(C_j|\boldsymbol{x}) \cong 0$.

Gaussian Class Conditionals

- Assume that the class-conditional densities are Gaussians and all classes share the same covariance matrix:

$$p(\boldsymbol{x} \mid C_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right\}$$

- We consider the case of two classes:

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a), \; a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)p(C_1)}{p(\boldsymbol{x} \mid C_2)p(C_2)}$$

where:

$$a(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_2) + \ln \frac{p(C_1)}{p(C_2)} =$$

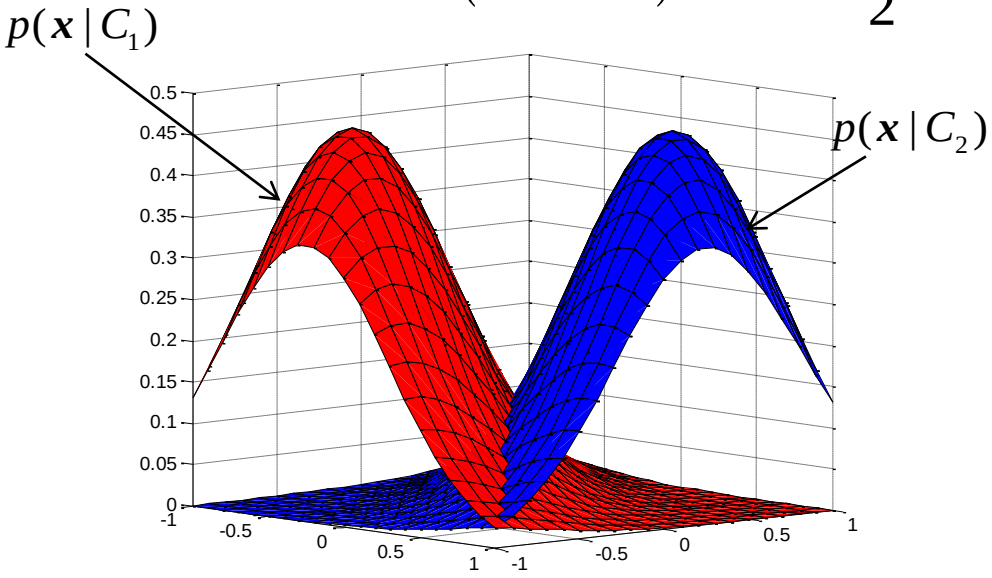
$$= \underbrace{\left(\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} - \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\right)}_{\boldsymbol{w}^T} \boldsymbol{x} \underbrace{- \frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(C_1)}{p(C_2)}}_{w_0}$$

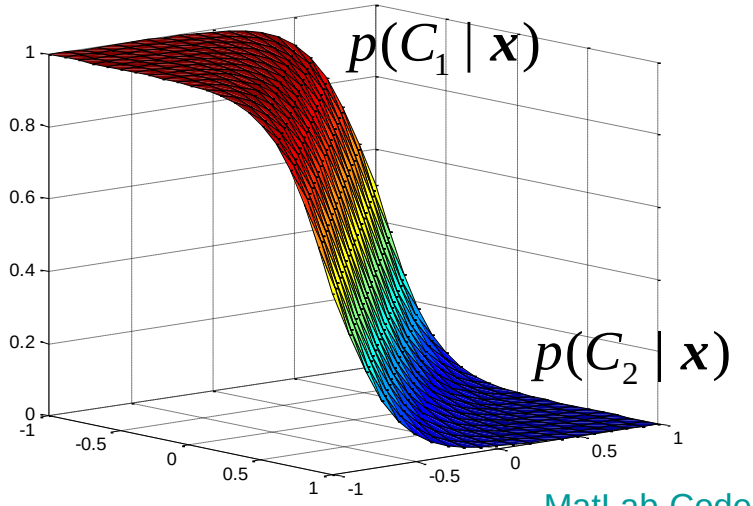
Gaussian Class Conditionals and Posteriors

- We thus obtain a linear function of $\boldsymbol{x}$:

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right),\ w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)}$$





MatLab Code

- Class conditional densities (left) and corresponding posteriors (right).
- On the right, the surface is colored using a proportion of red ink given by $p(C_1|\boldsymbol{x})$ and a proportion of blue ink given by $p(C_2|\boldsymbol{x}) = 1 - p(C_1|\boldsymbol{x})$

Gaussian Class Posteriors

$$p(C_1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x} + w_0)$$

- *The decision boundaries correspond to surfaces along which the posterior probabilities $p(C_k|\boldsymbol{x})$ are constant*.

- They are linear in input space.

- The prior probabilities $p(C_k)$ enter only through the bias parameter w_0

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(C_1)}{p(C_2)}$$

so *changing $p(C_k)$ only leads to parallel shifts of the decision boundaries.*

Probabilistic Generative Models K>2

- For the case of $K > 2$ classes, we have

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)\, p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)\, p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{where} \quad a_k = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right)$$

- The resulting decision boundaries, corresponding to the minimum misclassification rate, occur when the two largest posterior probabilities are equal, and so are defined by linear functions of $\boldsymbol{x}$, leading to a generalized linear model.

- For this case, we can see immediately that:

$$a_k(\boldsymbol{x}) = \ln\left(p(\boldsymbol{x} \mid C_k)\, p(C_k) \right) = -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{x} + \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

$$\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k, \; w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(C_k) + \ln \frac{1}{(2\pi)^{D/2}} - \frac{1}{2} \ln |\boldsymbol{\Sigma}|$$

Probabilistic Generative Models K>2

- Note that we can now re-write $p(C_k|\boldsymbol{x})$ by cancelling out terms:

$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)\, p(C_k)}{\sum_j p(\boldsymbol{x} \mid C_j)\, p(C_j)} = \frac{e^{a_k}}{\sum_j e^{a_j}} \quad \text{where} \quad a_k \equiv \boldsymbol{w}_k^T \boldsymbol{x} + w_{k0}$$

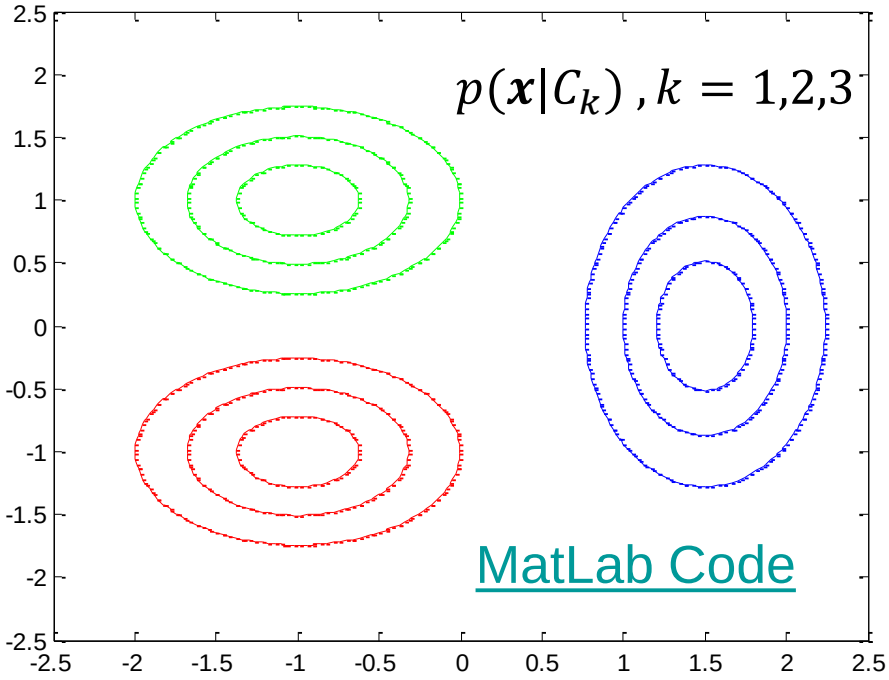
where:

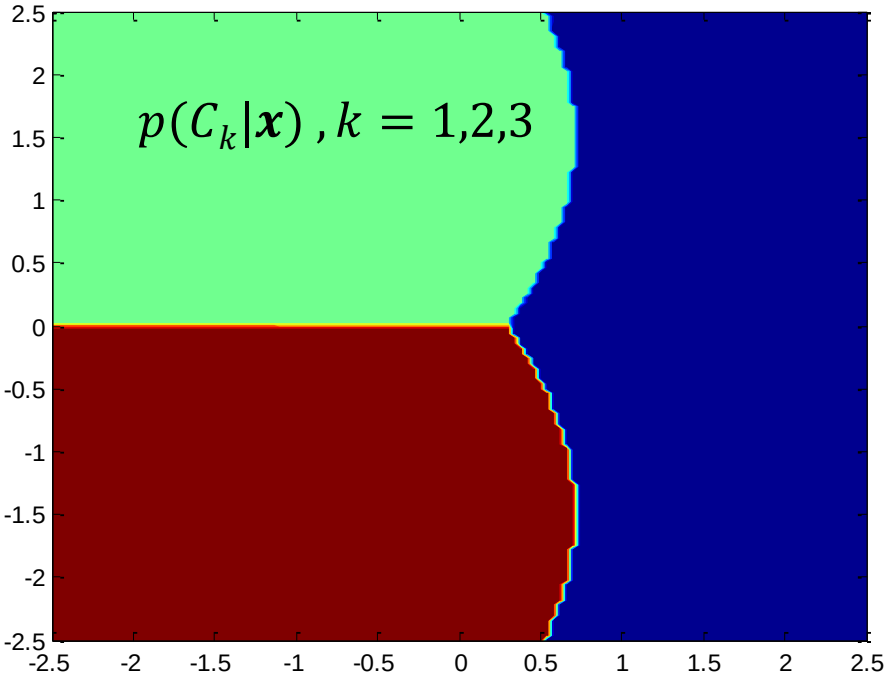
$$\boldsymbol{w}_k = \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k \,,\; w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \ln p(C_k)$$

- $a_k(\boldsymbol{x})$ are linear functions of $\boldsymbol{x}$ due to the cancellation of the quadratic terms with the shared $\boldsymbol{\Sigma}$.

Probabilistic Generative Models $K > 2$

- If we relax the assumption of a shared covariance matrix and allow each $p(\boldsymbol{x}|C_k)$ to have its own $\boldsymbol{\Sigma}_k$, then we obtain quadratic functions of $\boldsymbol{x}$ and a quadratic discriminant.

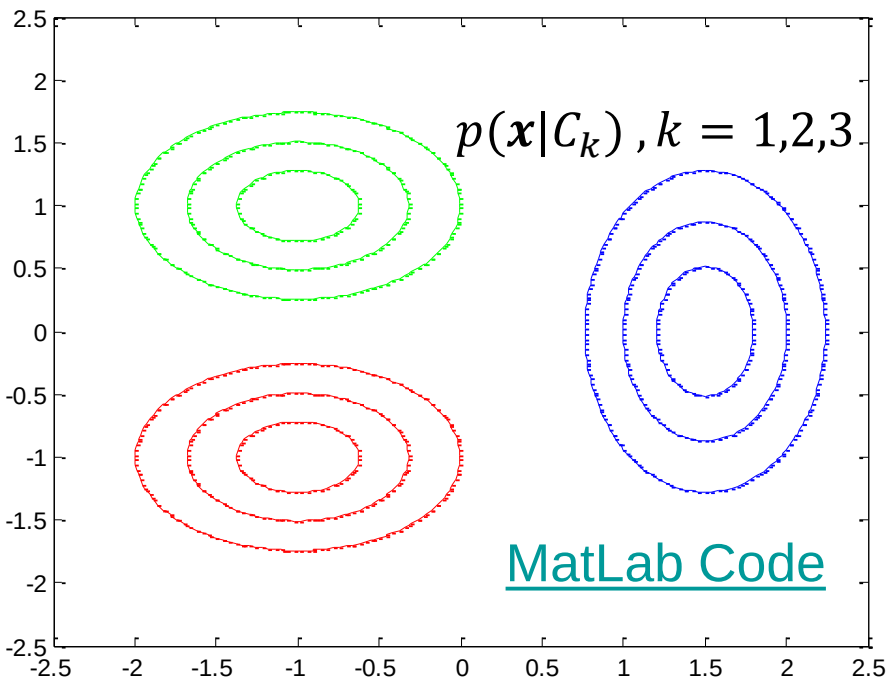


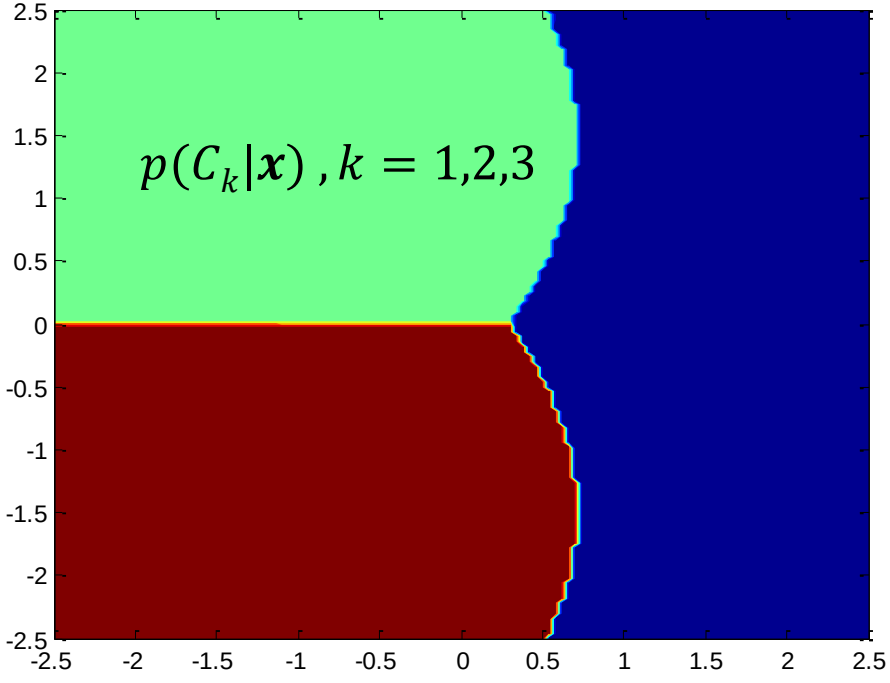


- The left-hand plot shows $p(\boldsymbol{x}|C_k), k = 1,2,3$ each a Gaussian distribution. The red & green classes have the same $\boldsymbol{\Sigma}$.

Probabilistic Generative Models $K > 2$

- The right-hand plot shows the posteriors $p(C_k|\boldsymbol{x})$, $k = 1,2,3$, in which the three colors represent the posterior probabilities for the respective 3 classes (scaled).
- Note that for *the two classes with the same covariance, the decision boundary is linear while for the other 2 is quadratic*.





Maximum Likelihood Solution

- Once we have a parametric form for $p(\boldsymbol{x}|C_k)$, we can compute the parameters, together with the prior class probabilities $p(C_k)$, using MLE.
- This requires a training data set comprising of $\boldsymbol{x}$ and their corresponding class labels.
- Consider the case of two classes each having a Gaussian class-conditional density with a shared covariance matrix.
- Suppose we have a data set $\{\boldsymbol{x}_n, t_n\}$ where $n = 1, \ldots, N$. Here $t_n = 1$ denotes class C_1 and $t_n = 0$ denotes C_2.
- We denote the prior class probabilities $p(C_1) = \pi$, $p(C_2) = 1 - \pi$.

Maximum Likelihood Solution

- For a data point $\boldsymbol{x}_n$ from class C_1, we have $t_n = 1$ and hence

$$p(\boldsymbol{x}_n, C_1) = p(C_1)\, p(\boldsymbol{x}_n \mid C_1) = \pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})$$

- Similarly for class C_2, we have $t_n = 0$

$$p(\boldsymbol{x}_n, C_2) = p(C_2)\, p(\boldsymbol{x}_n \mid C_2) = (1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})$$

- Thus *the likelihood function is given by*

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma})\right]^{t_n} \left[(1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma})\right]^{1-t_n}$$

where $\boldsymbol{t} = (t_1, \ldots, t_N)^T$.

- We maximize the log likelihood function.

Maximum Likelihood Solution

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider first the maximization with respect to π. The terms of interest in the log-likelihood are:

$$\sum_{n=1}^{N} \{ t_n \ln \pi + (1-t_n) \ln(1-\pi) \}$$

- Taking derivative wrt π and setting it equal to zero:

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N_1 + N_2}$$

where N_i the total number of data points in class C_i.

- As expected, the maximum likelihood estimate for π is simply the fraction of points in class C_1.
- This result is easily generalized to the multiclass case (see next) where the MLE of the prior probability associated with class C_k is given by the fraction of the training set points N_k.

Maximum Likelihood Solution: Multiclass

$$p(\{\boldsymbol{\phi}_n, \boldsymbol{t}_n\} \mid \{\pi_k\}) = \prod_{n=1}^{N} \prod_{k=1}^{K} \left[\pi_k \, p(\boldsymbol{\phi}_n \mid C_k)\right]^{t_{nk}}$$

- For the multiclass case, the log likelihood takes the form:

$$\ln p(\{\boldsymbol{\phi}_n, \boldsymbol{t}_n\} \mid \{\pi_k\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\ln \pi_k + \ln p(\boldsymbol{\phi}_n \mid C_k)\right), \quad \sum_{k=1}^{K} \pi_k = 1$$

- To maximize the log likelihood, we consider:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left(\ln \pi_k + \ln p(\boldsymbol{\phi}_n \mid C_k)\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1\right)$$

- Taking the derivative wrt π_k we arrive at the same result as for two classes:

$$\sum_{n=1}^{N} \frac{t_{nk}}{\pi_k} + \lambda = 0 \Rightarrow \pi_k \lambda = -\sum_{n=1}^{N} t_{nk} = -N_k \Rightarrow \begin{cases} \sum_{k=1}^{K} \pi_k \lambda = -N \\ \pi_k \lambda = -N_k \end{cases} \Rightarrow \begin{cases} \lambda = -N \\ \pi_k \lambda = -N_k \end{cases} \Rightarrow \pi_k = \frac{N_k}{N}$$

Maximum Likelihood Solution

$$p(\boldsymbol{t}, \boldsymbol{X} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider now the maximization with respect to μ_1. The terms of interest in the log-likelihood are:

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2} \sum_{n=1}^{N} t_n (\boldsymbol{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_1) + const.$$

- Taking derivative wrt μ_1 and setting it equal to zero:

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \boldsymbol{x}_n, \text{ and similarly } \boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1-t_n) \boldsymbol{x}_n$$

- Thus the MLE for μ_i is simply the mean of the points in class C_i.
- This result is easily generalized to the multiclass case. The MLE of the prior probability associated with class C_k is given by the fraction of the training set points N_k.

Maximum Likelihood Solution

$$p(\mathbf{t} \mid \pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} \left[\pi \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}) \right]^{t_n} \left[(1-\pi) \mathcal{N}(\mathbf{x}_n \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) \right]^{1-t_n}$$

- Consider finally the maximization with respect to $\boldsymbol{\Sigma}$. The terms of interest in the log-likelihood are:

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$-\frac{1}{2}\sum_{n=1}^{N} (1-t_n) \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} (1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_2)$$

$$= -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2} Tr\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{S}\right), \text{ where } \boldsymbol{S} = \frac{N_1}{N}\boldsymbol{S}_1 + \frac{N_2}{N}\boldsymbol{S}_2$$

$$\boldsymbol{S}_1 = \frac{1}{N_1}\sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \quad \boldsymbol{S}_2 = \frac{1}{N_2}\sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- Using the result for the MLE for a Gaussian distribution, we obtain $\boldsymbol{\Sigma} = \boldsymbol{S}$. This result is a weighted average of the covariance matrices associated with each class separately.

Maximum Likelihood Solution

$$\pi = \frac{N_1}{N_1 + N_2}, \ \boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n, \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n)\mathbf{x}_n$$

$$\boldsymbol{\Sigma} = \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2, \ \mathbf{S}_1 = \frac{1}{N_1}\sum_{n \in C_1} (\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T, \mathbf{S}_2 = \frac{1}{N_2}\sum_{n \in C_2} (\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T$$

- These results are easily extended to the K class problem to obtain the corresponding MLE solutions for the parameters (for Gaussian class-conditionals with a shared covariance matrix).

- As we have seen before, MLE is not robust to outliers.

  - Fitting Gaussian distributions to the classes is not robust to outliers, because the MLE of a Gaussian is not robust.

Discrete Features: Naive Bayes Assumption

- Here we consider the case of discrete feature values x_i. Consider binary feature values $x_i \in \{0, 1\}, i = 1, \ldots, D$.
- For D inputs, a general distribution corresponds to a table of 2^D numbers for each class, containing $2^D - 1$ independent variables (due to the summation constraint).
- This grows exponentially with the number D of features.
- Consider the Naive Bayes Assumption: the feature values are treated as independent, conditioned on the class C_k.
- Thus we have class-conditional distributions of the form

$$p(\mathbf{x} \mid C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1 - \mu_{ki})^{1 - x_i}$$

Discrete Features

$$p(\boldsymbol{x} \mid C_k) = \prod_{i=1}^{D} \mu_{ki}^{x_i} (1-\mu_{ki})^{1-x_i}$$

- There are D independent parameters for each class.
- Here μ_{ki} is the probability the i^{th} feature takes the value 1.
- Substituting into $a_k = \ln\big(p(\boldsymbol{x} \mid C_k)\, p(C_k)\big)$ gives:

$$a_k(\boldsymbol{x}) = \sum_{i=1}^{D} \big\{ x_i \ln \mu_{ki} + (1-x_i) \ln (1-\mu_{ki}) \big\} + \ln p(C_k)$$

- These are linear functions of the input values x_i.
- For $K = 2$, we can alternatively consider the logistic sigmoid

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a), \; a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1)\, p(C_1)}{p(\boldsymbol{x} \mid C_2)\, p(C_2)}$$

- Analogous results are obtained for discrete variables each of which can take $M > 2$ states.

Exponential Family

- We have have seen that for both Gaussian and discrete inputs, the posterior class probabilities are given by generalized linear models with logistic sigmoid ($K = 2$) or softmax ($K \geq 2$ classes) activation functions.

- This result is general for the class-conditional densities $p(\boldsymbol{x}|C_k)$ that are members of the exponential family.

- For the exponential family, the distribution of $\boldsymbol{x}$ can be written in the form

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k) = h(\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\boldsymbol{\lambda}_k^T u(\boldsymbol{x})\right\}$$

- Here, we consider the case $u(\boldsymbol{x}) = \boldsymbol{x}$.

Exponential Family

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k) = h(\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\boldsymbol{\lambda}_k^T u(\boldsymbol{x})\right\}$$

- With $u(\boldsymbol{x}) = \boldsymbol{x}$ and by introducing a scaling parameter s as $p(x \mid s) = (1/s) f(x/s)$, we obtain the restricted set of exponential family class-conditional densities of the form:

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k, s) = \frac{1}{s} h(\frac{1}{s}\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x}\right\}$$

- Here each class has its own parameter λ_k but all classes share the same scale parameter s.

- For the 2 class problem, we substitute this expression for the class-conditional densities into

$$a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1) p(C_1)}{p(\boldsymbol{x} \mid C_2) p(C_2)}$$

Exponential Family

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k, s) = \frac{1}{s} h(\frac{1}{s}\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x}\right\} \qquad a(\boldsymbol{x}) = \ln \frac{p(\boldsymbol{x} \mid C_1) p(C_1)}{p(\boldsymbol{x} \mid C_2) p(C_2)}$$

- The posterior class probability is again given by a logistic sigmoid acting on a linear function $a(\boldsymbol{x})$

$$p(C_1 \mid \boldsymbol{x}) = \sigma(a(\boldsymbol{x}))$$

which is given by

$$a(\boldsymbol{x}) = \frac{1}{s}\left(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\right)^T \boldsymbol{x} + \ln g(\boldsymbol{\lambda}_1) - \ln g(\boldsymbol{\lambda}_2) + \ln p(C_1) - \ln p(C_2)$$

- For the K –class problem, we substitute the class-conditionals

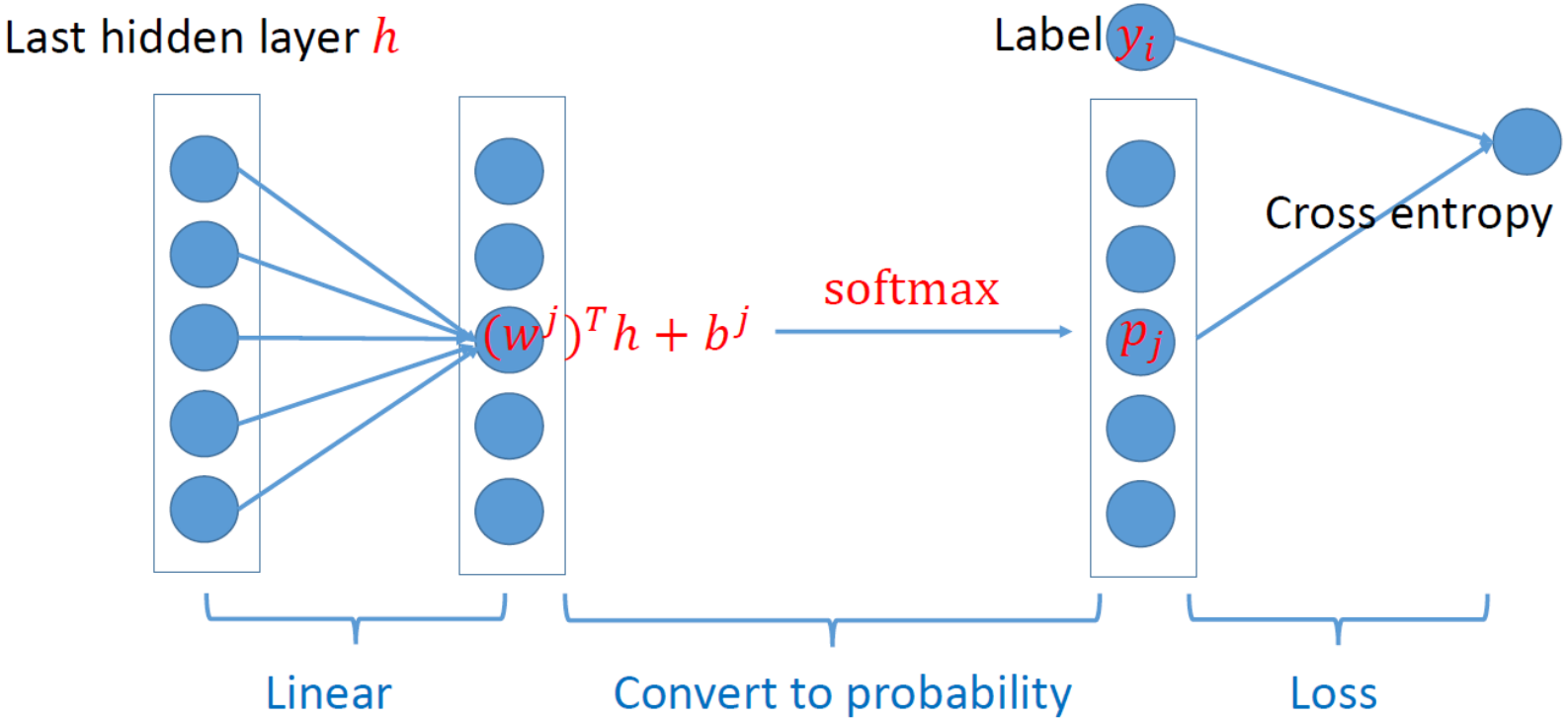
directly in $a_k(\boldsymbol{x}) = \ln\left(p(\boldsymbol{x} \mid C_k) p(C_k)\right)$ and $p(C_k \mid \boldsymbol{x}) = \frac{e^{a_k}}{\sum_j e^{a_j}}$ to obtain:

$$a_k(\boldsymbol{x}) = \frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x} + \ln g(\boldsymbol{\lambda}_k) + \ln p(C_k)$$

which is linear in $\boldsymbol{x}$.

Cross Entropy for Conditional Distribution

- Let $p_{data}(y|\boldsymbol{x})$ denote the empirical distribution of the data.
- The negative log-likelihood: $-\frac{1}{N}\sum_{i=1}^{N} logp(y = y_i|\boldsymbol{x}_i) = -\mathbb{E}_{p_{data}} logp(y|\boldsymbol{x})$ is the cross entropy between p_{data} and the model output p.
- Write this as KL minimization: $D(p_{data}||p) = \mathbb{E}_{p_{data}}[logp_{data}] - \mathbb{E}_{p_{data}}[logp]$. The 1st term (entropy) is const. & the 2nd the cross entropy.



Stochastic Gradient Descent
Robbins-Monro Algorithm

Stochastic Gradient Descent

- **Stochastic gradient descent** is a gradient descent optimization method for minimizing an objective function written as a sum of differentiable functions.

- Consider minimizing an objective function that has the form of a sum (M –estimators)

$$f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{f(\boldsymbol{\theta}, \boldsymbol{z}_i)}_{Loss\ Function}, \boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$$

- The parameter θ is to be estimated and each summand function is related with the i –th observation in the data set (used for training).

  - Typical problems are the maximization (stationary points) of the likelihood function, ,

$$f(\boldsymbol{\theta}, z_i) = -\log p(y_i \mid \boldsymbol{x}_i, \boldsymbol{\theta})$$

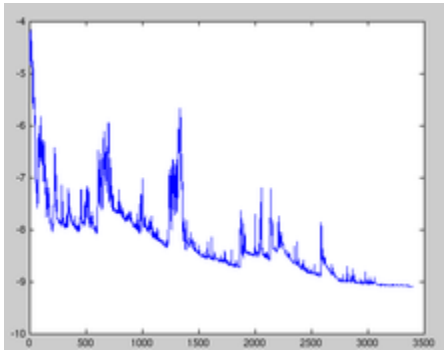
  - Minimization of a loss function, etc.

$$f(\boldsymbol{\theta}, \boldsymbol{z}_i) = L\big(y_i, \boldsymbol{h}(\boldsymbol{x}_i, \boldsymbol{\theta})\big) = L(y_i, \hat{y}_i),\ \hat{y}_i \equiv h(\boldsymbol{x}_i, \theta)$$

Stochastic Gradient Descent

- A batch gradient descent method performs the following iterations

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \sum_{i=1}^{N} \nabla f\left(\boldsymbol{\theta}, \mathbf{z}_i\right)$$



- η is the *learning rate*. Fluctuations in the objective function with gradient steps are shown here
- Evaluating the sums of gradients is expensive.
- Stochastic gradient descent samples a subset of summand functions at every step. This is very effective in the case of large-scale machine learning problems.
- For example passing through each data point in the training set:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla f\left(\boldsymbol{\theta}, \mathbf{z}_i\right)$$

- Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Intl. Conf. on Machine Learning*, pp. 928-936.
- Cesa-Bianchi, N. and G. Lugosi (2006). *Prediction, learning, and games*. Cambridge University Press.
- Nemirovski, A. and D. Yudin (1978). On Cezari's convergence of the steepest descent method for approximating saddle points of convex-concave functions. *Soviet Math. Dokl. 19*.
- Sra et al. Optimization for Machine Learning, 2012.

Stochastic Gradient Descent

- One can use a running average of the parameter

$$\bar{\theta}_k = \frac{1}{k}\sum_{i=1}^{k}\theta_i$$

- Or use the Polyak-Ruppert averaging

$$\bar{\theta}_k = \bar{\theta}_{k-1} - \frac{1}{k}\left(\theta_{k-1} - \theta_k\right)$$

- Spall, J. (2003). *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley.
- Kushner, H. and G. Yin (2003). *Stochastic approximation and recursive algorithms and applications*. Springer.

Stochastic Optimization

- Stochastic optimization algorithms attempt to find zeroes or extrema of functions which cannot be computed directly, but only estimated via noisy observations. Mathematically, this refers to solving:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) = \mathbb{E}\left[f(\boldsymbol{\theta}, \boldsymbol{z}) \right]$$

- The objective is to find the parameter $\boldsymbol{\theta}$, which minimizes $f(\boldsymbol{\theta})$ *for some unknown random variable,* $\boldsymbol{z}$.
- $f(\boldsymbol{\theta})$ *cannot be computed exactly, but instead approximated via simulation*. This situation is modeled by the function $f(\boldsymbol{\theta}, \boldsymbol{z})$, where $\boldsymbol{z}$ represents the noise and is a random variable.
- The objective is then to minimize the expectation of $f(\boldsymbol{\theta}, \boldsymbol{z})$. Our earlier objective function can be considered as a sampled approximation of the expectation.
- The first algorithm of this kind is the Robbins-Monro algorithm.

- Robbins H., Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics* **22** (3): 400.

Robbins Monro Algorithm

- Robbins–Monro algorithm first introduced to solve a root finding problem where the function is represented as expected value.
- Assume that we have a function $M(\theta)$, and a constant α, such that the equation $M(\theta) = \alpha$ has a unique root at $\theta = \theta^*$.
- It is assumed that while we cannot directly observe the function $M(\theta)$, we can instead obtain measurements of the random variable $N(\theta)$ where $\mathbb{E}[N(\theta)] = M(\theta)$. The structure of the algorithm is to then generate iterates of the form:

$$\theta_{n+1} = \theta_n - \eta_n \left(N\left(\theta_n\right) - \alpha \right), \eta_n \; sequence \; of \; > 0 \; stepsizes$$

- The sequence θ_n converges in L_2 to θ^*, provided that:
  - $N(\theta)$ is uniformly bounded
  - $M(\theta)$ is nondecreasing
  - $M'(\theta^*)$ exists and is positive
  - The following hold: $\sum_{n=0}^{\infty} \eta_n = \infty, \sum_{n=0}^{\infty} \eta_n^2 < \infty$
- A particular choice is: $\eta_n = a/n$, for some $a > 0$.

Robbins Monro Algorithm

- Another choice of the learning rate schedule is:

$$\eta_k = \frac{1}{(\tau_0 + k)^{\kappa}}, \tau_0 \geq 0, \kappa \in (0.5, 1]$$

- The need to adjust these tuning parameters is one of the main drawback of stochastic optimization.

- A simple approach is as follows:
  - store an initial subset of the data, and try a range of η values on this subset;
  - then choose the one that results in the fastest decrease in the objective and apply it to all the rest of the data.
  - Convergence is not certain but the algorithm can be terminated when the performance improvement on a hold-out set plateaus (early stopping).

- Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (Ed.), *Online Learning and Neural Networks*. Cambridge (presentations)
- Bach, F. and E. Moulines (2011). Nonasymptotic analysis of stochastic approximation algorithms for machine learning. In *NIPS*.
- Bottou, L. (2007). Learning with large datasets (nips tutorial).

Per-Parameter Step Sizes

- One drawback of SGD is that it uses the same step size for all parameters. This can be extended using a method known as **adagrad** (adaptive gradient).

- In particular, if $\theta_i(k)$ is parameter θ_i at time k, and $g_i(k)$ is its gradient, then we make an update as follows:

$$\theta_i(k+1) = \theta_i(k+1) - \eta \frac{g_i(k)}{\tau_0 + \sqrt{s_i(k)}}, \; s_i(k) = s_i(k-1) + g_i(k)^2$$

- Since $s_i(k)$ is the ℓ_2 norm of previous derivatives, extreme parameter updates get dampened, while parameters that get few or small updates receive higher learning rates.

- The result is *a per-parameter step size that adapts to the curvature of the loss function.*

- Duchi, J., E. Hazan, and Y. Singer (2010). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *Proc. of the Workshop on Computational Learning Theory*
- Schaul, T., S. Zhang, and Y. LeCun (2012). No more pesky learning rates. Technical report, Courant Institute of Mathematical Sciences.

The LMS Algorithm For Regression

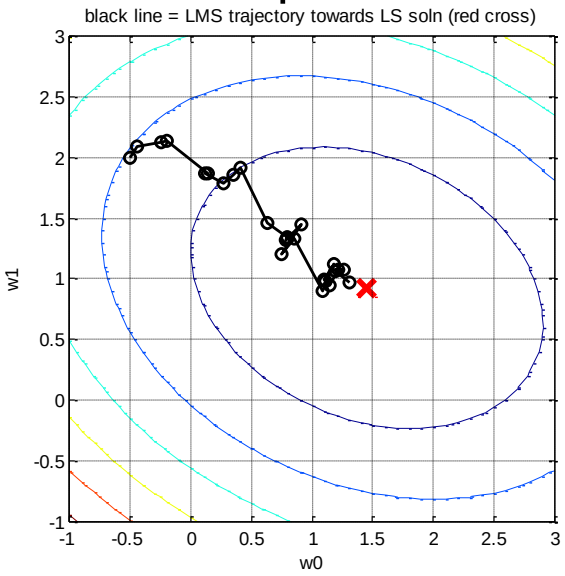
- Let us recall the MLE for linear regression in an online fashion (see earlier lecture on Regression). The online gradient at iteration k is

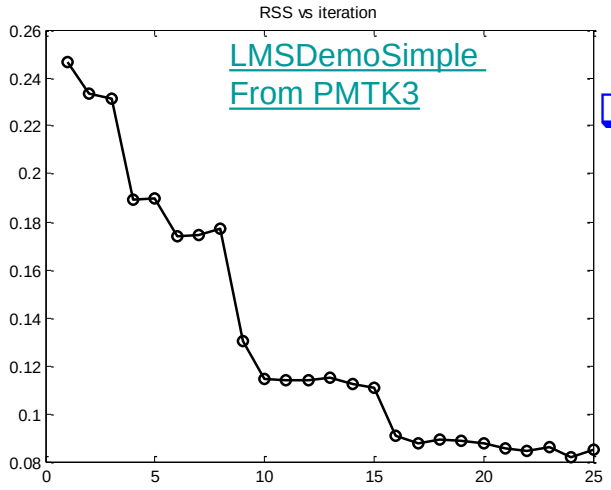
$$\boldsymbol{g}_k = \boldsymbol{x}_i\left(\boldsymbol{\theta}_k^T \boldsymbol{x}_i - y_i\right)$$

- The gradient acts like an error signal. After computing the gradient, we take a step along it as follows:

$$\boldsymbol{\theta}_{(k+1)} = \boldsymbol{\theta}_{(k)} - \eta_k(\hat{y}_k - y_k)\boldsymbol{x}_k,\ \hat{y}_k = \boldsymbol{\theta}_k^T \boldsymbol{x}_k$$

- This algorithm is called the least mean squares (LMS) and is an example of the SGD algorithm.





- LMS may require multiple passes through the data to find the optimum.

- *The recursive least squares algorithm*, based on the Kalman filter and which uses 2nd-order information, finds the optimum in a single pass (see Murphy, Section 18.2.3)

The Perceptron Algorithm

The Rosenblatt Perceptron Algorithm

- This is a two-class model in which $\boldsymbol{x}$ is mapped to a feature $\phi(\boldsymbol{x})$, that is used to construct a generalized linear model

$$y(\boldsymbol{x}) = f(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))$$

where the activation function f is: $f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$

- *The vector $\boldsymbol{\phi}(\boldsymbol{x})$ includes a bias component $\phi_0(\boldsymbol{x}) = 1$.*
- We use target $t = +1$ *for class* C_1 and $t = -1$ for class C_2.
- When using as error function the total number of misclassified patterns, the error is a piecewise constant function of $\boldsymbol{w}$, with discontinuities wherever a change in $\boldsymbol{w}$ causes the decision boundary to move across one of the data points.
  - Gradient optimization methods cannot be applied, because the gradient is zero almost everywhere.

- Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.

The Rosenblatt Perceptron Algorithm

- We consider an alternative error function (perceptron criterion).
- We are seeking a weight vector $\boldsymbol{w}$ such that $\boldsymbol{x}_n$ in class C_1 will have $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) > 0$, whereas $\boldsymbol{x}_n$ in C_2 have $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) < 0$.
- Using the $t \in \{1, -1\}$ target coding scheme it follows that we would like all patterns to satisfy $\boldsymbol{w}^T\phi(\boldsymbol{x}_n)\, t_n > 0$.
- *The perceptron associates zero error with a correctly classified pattern, whereas for a misclassified $\boldsymbol{x}_n$ it minimizes $-\boldsymbol{w}^T\phi(\boldsymbol{x}_n)\, t_n$.*

- The perceptron criterion is

$$E_P(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} \boldsymbol{w}^T \boldsymbol{\phi}_n t_n$$

$\mathcal{M}$ is the set of all misclassified patterns.

The Rosenblatt Perceptron Algorithm

- The contribution to the error from a misclassified pattern is a linear in $\boldsymbol{w}$ in regions of $\boldsymbol{w}$ -space where the pattern is misclassified and zero in regions where it is correctly classified.
- The total error function is piecewise linear

$$E_P(\boldsymbol{w}) = -\sum_{n \in \mathcal{M}} \boldsymbol{w}^T \boldsymbol{\phi}_n t_n$$

- We apply the stochastic gradient descent algorithm:

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla E_P(\boldsymbol{w}) = \boldsymbol{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n$$

where η is the learning rate parameter.

- Since the perceptron function $y(\boldsymbol{x}, \boldsymbol{w})$ is unchanged if we multiply $\boldsymbol{w}$ by a const., we set $\eta = 1$ without loss of generality.
  - During training, the set of patterns that are misclassified will change.

The Rosenblatt Perceptron Algorithm

- The perceptron learning algorithm has a simple interpretation

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \boldsymbol{\phi}_n t_n$$

- We cycle through the training patterns, and for each $\boldsymbol{x}_n$ we evaluate the perceptron function $y(\boldsymbol{x}) = f(\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}))$.

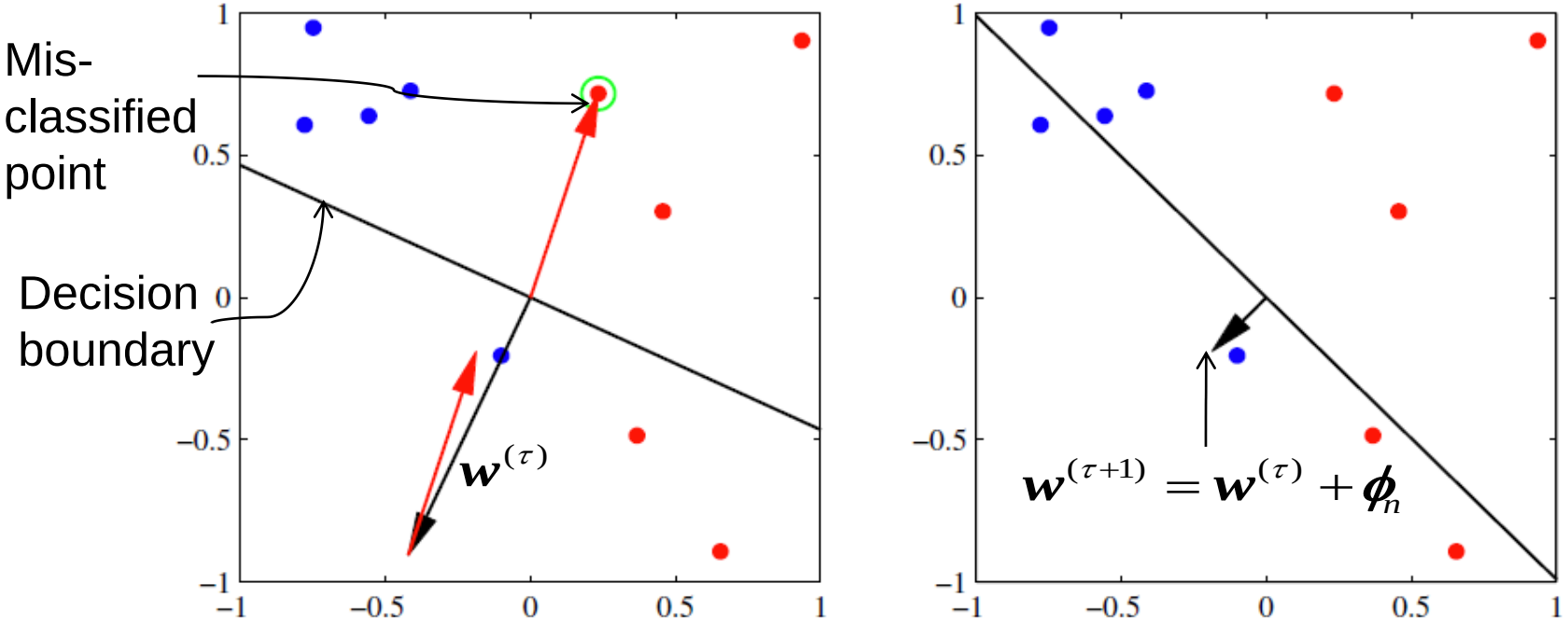
$$f(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

- If the pattern is correctly classified, then the weight vector remains unchanged.

- If the pattern is incorrectly classified, then for class C_1 we add $\boldsymbol{\phi}(\boldsymbol{x}_n)$ onto the current estimate of weight vector $\boldsymbol{w}$ while for C_2 we subtract $\boldsymbol{\phi}(\boldsymbol{x}_n)$ from $\boldsymbol{w}$.

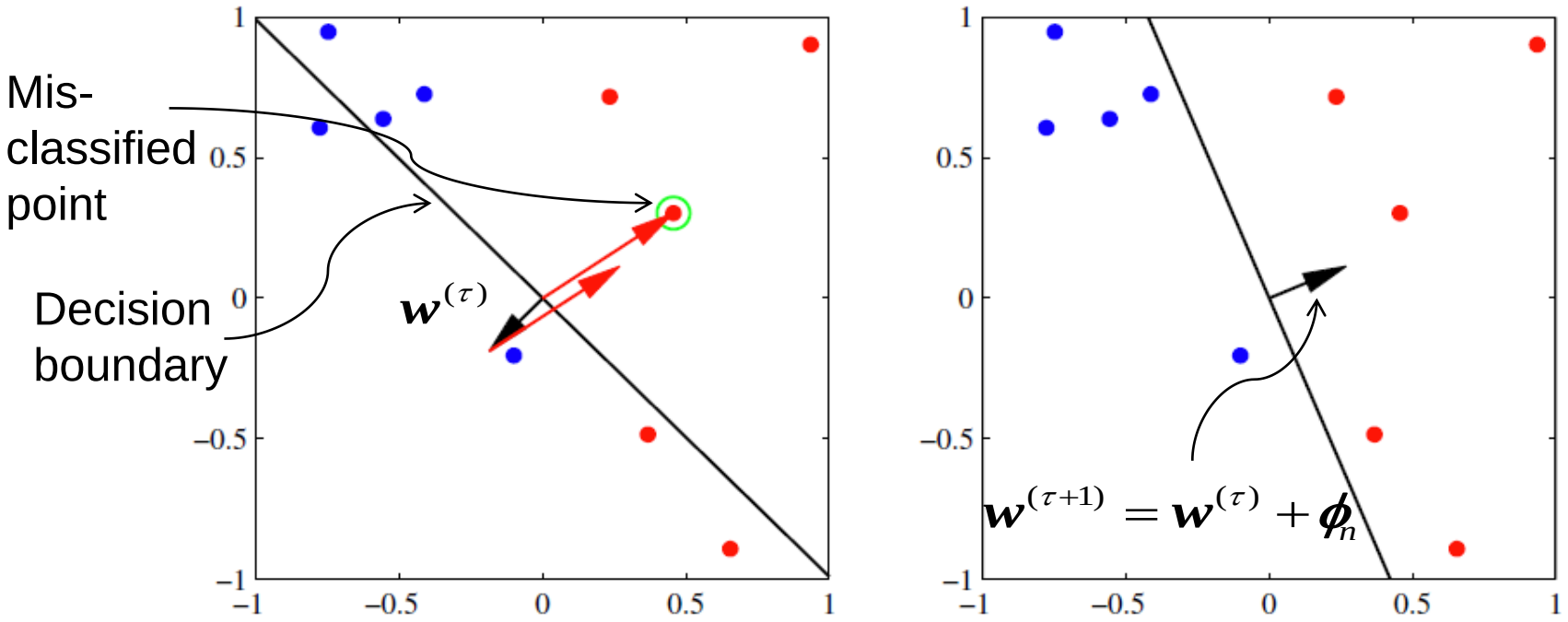
The Rosenblatt Perceptron Algorithm

- Data points from two classes (red and blue) in a feature space (ϕ_1, ϕ_2).
- The parameter vector $\boldsymbol{w}^{(\tau)}$ points towards the red class.
- The data point shown is misclassified and so its feature vector is added to the current weight vector, giving the new decision boundary on the right.



The Rosenblatt Perceptron Algorithm

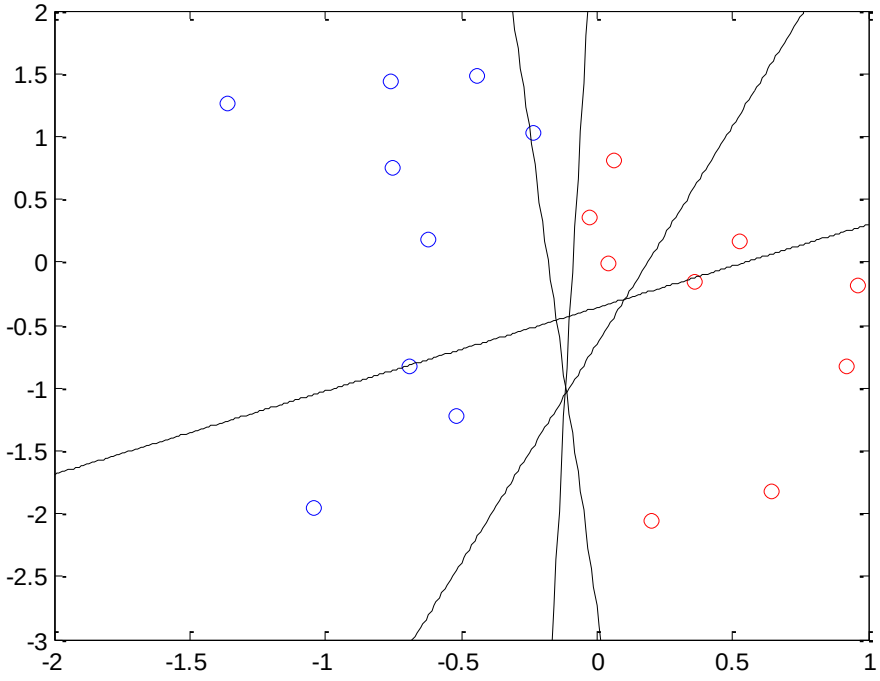
- The left plot shows the next misclassified point.
- Its feature vector is added to the weight vector giving the decision boundary shown in the right plot for which all data points are correctly classified.

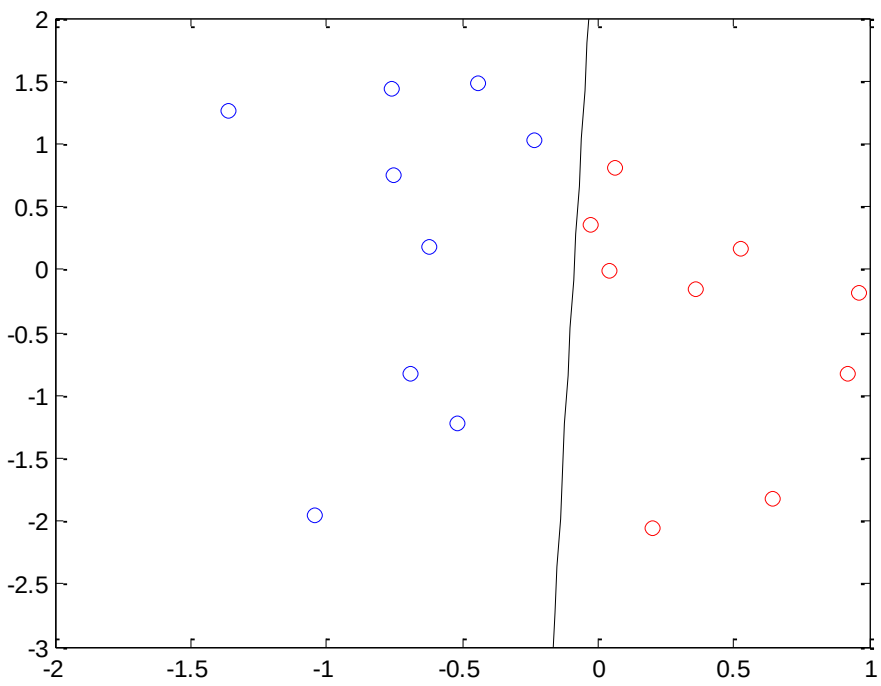


MatLab Code

The Rosenblatt Perceptron Algorithm

- Demonstration of the algorithm using the data set *data_RP_10* (very fast convergence)

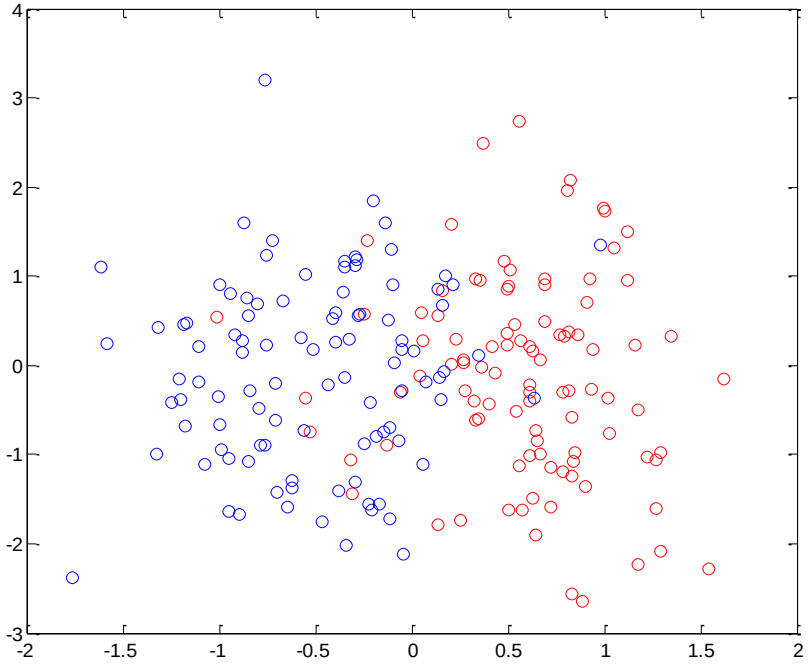


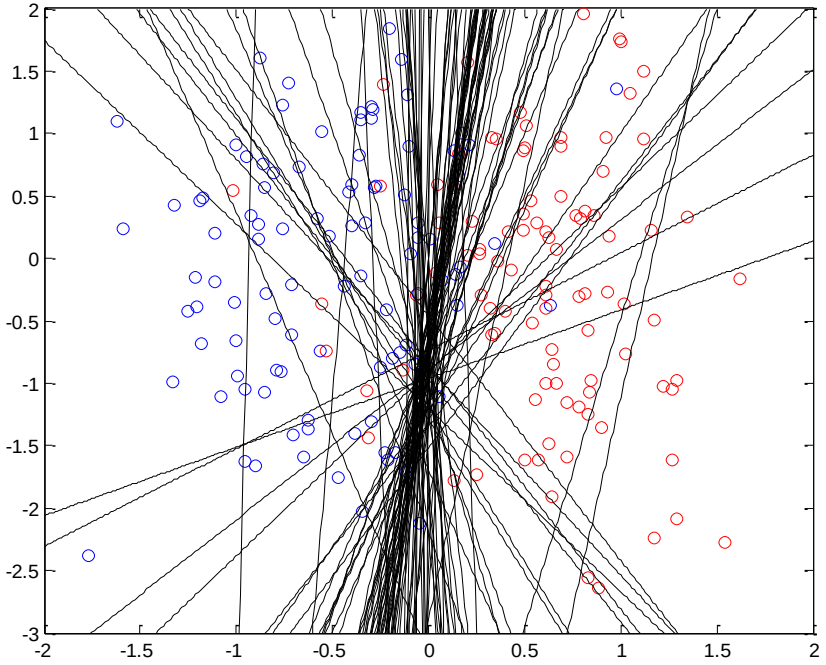


MatLab Code

The Rosenblatt Perceptron Algorithm

- Use data set *data_RP_100* (very slow convergence)
- Weakness: the algorithm will never converge if the two classes overlap, which means we cannot find a decision boundary that completely separates them.





MatLab Code

The Rosenblatt Perceptron Algorithm

- In each update in the perceptron learning algorithm, the contribution to the error from a misclassified pattern will be reduced because from $\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} + \eta \boldsymbol{\phi}_n t_n$ we have

$$-\boldsymbol{w}^{(\tau+1)T}\boldsymbol{\phi}_n t_n = -\boldsymbol{w}^{(\tau)T}\boldsymbol{\phi}_n t_n - \left(\boldsymbol{\phi}_n t_n\right)^T \boldsymbol{\phi}_n t_n < -\boldsymbol{w}^{(\tau)T}\boldsymbol{\phi}_n t_n$$

where we set $\eta = 1$.

- This does not imply that the contribution to the error from the other misclassified patterns will have been reduced.
- Also the change in $\boldsymbol{w}$ may have caused some previously correctly classified patterns to become misclassified.

The perceptron learning rule is not guaranteed to reduce the total error function at each stage

The Perceptron Convergence Theorem

- For linearly separable training data set, the perceptron learning algorithm is guaranteed to find an exact solution in a finite number of steps.
- The number of steps to achieve convergence could be substantial.
- Until convergence is achieved, one cannot distinguish a nonseparable problem and one that is slow to converge.
- There may be many solutions: initialization of the parameters and the order of going through the data points affects the solution obtained.

For data sets that are not linearly separable, the perceptron learning algorithm will never converge

The Perceptron Convergence Theorem

- The perceptron does not provide probabilistic outputs.
- It cannot be applied to $K > 2$ classes.
- Like with all other classification algorithms examined up to now, it is based on linear combinations of fixed basis functions.

- Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan.
- Block, H. D. (1962). The perceptron: a model for brain functioning. *Reviews of Modern Physics* **34**(1), 123–135. Reprinted in Anderson and Rosenfeld (1988).
- Nilsson, N. J. (1965). *Learning Machines*. McGraw- Hill. Reprinted as *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann, (1990).
- Minsky, M. L. and S. A. Papert (1969). *Perceptrons*. MIT Press. Expanded edition 1990.
- Hertz, J., A. Krogh, and R. G. Palmer (1991). *Introduction to the Theory of Neural Computation*. Addison Wesley.
- Bishop, C. M. (1995a). *Neural Networks for Pattern Recognition*. Oxford University Press.
- Widrow, B. and M. E. Hoff (1960). Adaptive switching circuits. In *IRE WESCON Convention Record*, Volume 4, pp. 96–104. Reprinted in Anderson and Rosenfeld (1988).
- Widrow, B. and M. A. Lehr (1990). 30 years of adaptive neural networks: perceptron, madeline, and backpropagation. *Proceedings of the IEEE* **78**(9), 1415–1442.

A Bayesian Perspective

- Another approach to online learning is to adopt a Bayesian view. We just apply Bayes rule recursively:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:k}) \propto p(\mathcal{D}_k|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1})$$

- This has the advantage of returning a posterior instead of just a point estimate. It also allows for the online adaptation of hyper-parameters.

- Finally, it is quicker than SGD. Note that *by modeling the posterior variance of each parameter, we effectively associate a different learning rate for each parameter, which is a simple way to model the curvature of the space.*

- These variances can then be adapted using the usual rules of probability theory.

- de Freitas, N., M. Niranjan, and A. Gee (2000). Hierarchical Bayesian models for regularisation in sequential learning. *Neural Computation 12*(4), 955–993.

A Bayesian Perspective

- Getting 2nd-order optimization methods to work online is tricky.[1-4]
- One can use the Kalman filter to fit a linear regression model online. Unlike the LMS algorithm, this converges to the optimal (offline) answer in a single pass over the data.
- An extension which can learn a robust non-linear regression model in an online fashion is described in [5]. For the GLM case, we can use an assumed density filter, where we approximate the posterior by a Gaussian with a diagonal covariance; the variance terms serve as a per-parameter step-size.
- Another approach is to use particle filtering [6] for sequentially learning a kernelized linear/logistic regression model.

1. Schraudolph, N. N., J. Yu, and S. Gunter (2007). A Stochastic Quasi- Newton Method for Online Convex Optimization. In *AI/Statistics*, pp. 436–443.
2. Sunehag, P., J. Trumpf, S. V. N. Vishwanathan, and N. N. Schraudolph (2009). Variable Metric Stochastic Approximation Theory. In *AI/Statistics*, pp. 560–566.
3. Bordes, A., L. Bottou, and P. Gallinari (2009, July). Sgd-qn: Careful quasi-newton stochastic gradient descent. *J. of Machine Learning Research 10*, 1737–1754.
4. Bordes, A., L. Bottou, P. Gallinari, J. Chang, and S. A. Smith (2010). Erratum: SGDQN is Less Careful than Expected. *J. of Machine Learning Research 11*, 2229–2240.
5. Ting, J., A. D'Souza, S. Vijayakumar, and S. Schaal (2010). Efficient learning and feature selection in high-dimensional regression. *Neural Computation 22*(4), 831–886.
6. Andrieu, C., N. de Freitas, and A. Doucet (2000). Sequential Bayesian estimation and model selection for dynamic kernel machines. Technical report, Cambridge Univ.

Probabilistic Discriminative Models

Probabilistic Discriminative Models

- *For the two-class problem*, *the posterior probability of class* C_k *can be written as a logistic sigmoid* acting on a linear function of $\boldsymbol{x}$, for a wide choice of $p(\boldsymbol{x}|C_k)$.
- *For the multiclass problem*, the posterior probability of C_k is given by *a softmax transformation* of a linear function of $\boldsymbol{x}$.
- *For specific choices of* $p(\boldsymbol{x}|C_k)$*, we used MLE to determine the parameters of the densities and the class priors* $p(C_k)$ *and then used Bayes' theorem to find the posterior class probabilities.*
- An alternative approach that we follow here is to *use the functional form of the generalized linear model explicitly and determine its parameters directly by using MLE*.
- There is an efficient algorithm finding such solutions known as iterative reweighted least squares, or *IRLS.*

Probabilistic Discriminative Models

- Indirect approach – Generative Modeling: find the parameters of the generalized linear model, by fitting class-conditional densities and class priors separately and then applying Bayes' theorem.
  - With this model, we *can generate synthetic data by drawing values of $\boldsymbol{x}$ from the marginal $p(\boldsymbol{x})$.*
- Direct approach – Discriminative Modeling: we are maximizing a likelihood function defined through $p(C_k|\boldsymbol{x})$, which represents a form of discriminative training.
- An advantage of the discriminative approach is that there are typically fewer adaptive parameters to be determined.
- It leads to improved predictive performance when the class-conditional density assumptions are a poor approximation to the true distributions.

Fixed Basis Functions

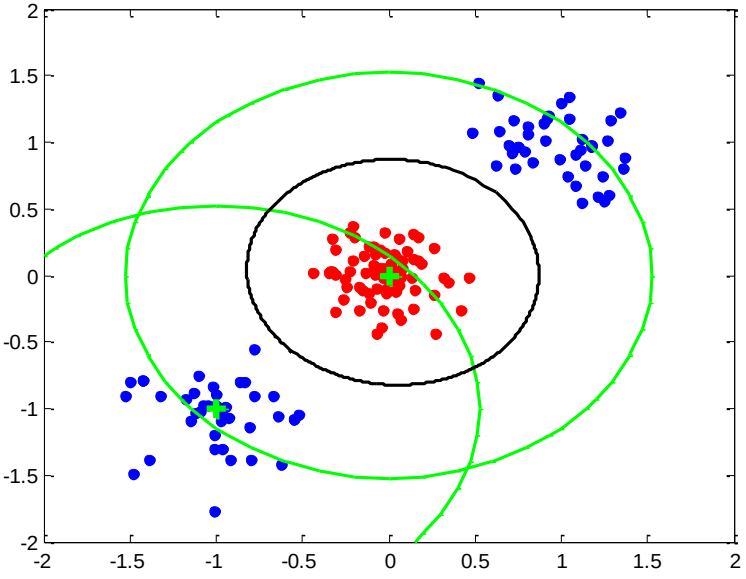
- Before we start with the logistic regression model, let us first make a fixed nonlinear transformation of the inputs using a vector of basis functions $\phi(\boldsymbol{x})$, then apply the algorithms discussed up to now.

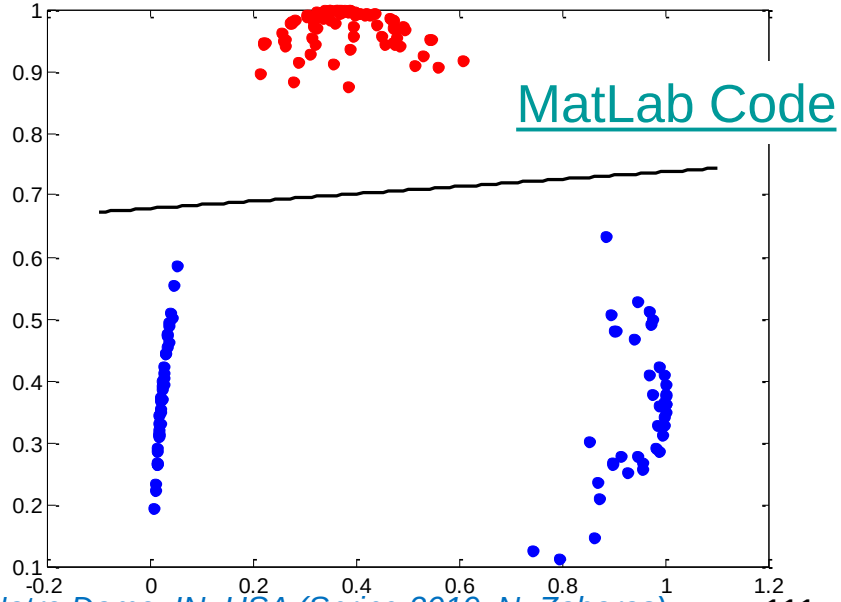
- The resulting decision boundaries will be linear in the feature space ϕ, and these correspond to nonlinear decision boundaries in the original $\boldsymbol{x}$ space.

- Classes that are linearly separable in the feature space $\phi(\boldsymbol{x})$ need not be linearly separable in the original input space $\boldsymbol{x}$.

Nonlinear Basis in Classification Models

- Left: original space (x_1, x_2) with data from red & blue classes
- Two 'Gaussian' basis functions $\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x})$ are defined in (x_1, x_2) with centers shown by the green crosses & contours by the green circles.
- Right: the corresponding feature space (ϕ_1, ϕ_2) together with the linear decision boundary obtained by a logistic regression model. This corresponds to a nonlinear decision boundary in (x_1, x_2) (black curve on the left).





MatLab Code

Fixed Basis Functions

- One of the basis functions is typically set to a constant, say $\phi_0(x) = 1$, so that w_0 plays the role of a bias.
- For problems with significant overlap between the class-conditional densities $p(\boldsymbol{x}|C_k)$, the posterior probabilities $p(C_k|\boldsymbol{x})$ at least for some values of $\boldsymbol{x}$ may not be 0 or 1.
  - In such cases, the optimal solution is obtained by modeling the posterior probabilities accurately and then applying standard decision theory.
- The transformation $\phi(\boldsymbol{x})$ cannot remove class overlap, it can increase the level of overlap, or create overlap where none existed in the $\boldsymbol{x}$ space.
- However, suitable choices of nonlinearity can make the process of modeling the posterior probabilities easier.

Fixed Basis Functions

- Fixed basis function models have serious limitations.
- We should allow the basis functions to adapt to the data.
- Still models with fixed nonlinear basis functions are important as they introduce many of the key concepts needed for an understanding models with data-adaptive basis.

Logistic Regression

- Consider a simple generalized linear problem with a two-class classification.

- In generative models we saw that the posterior probability of class C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector $\boldsymbol{\phi}$ so that

$$p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

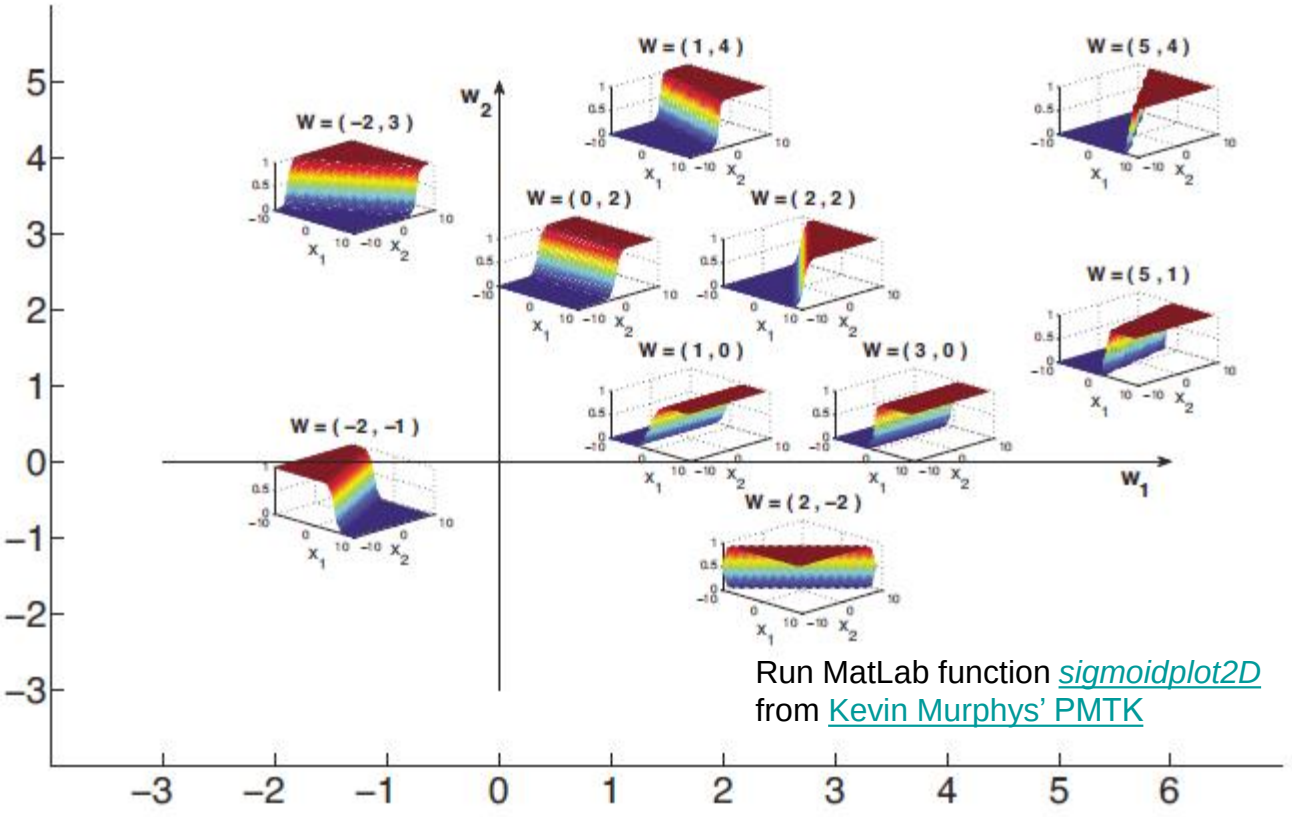
with $p(C_2|\boldsymbol{\phi}) = 1 - p(C_1|\boldsymbol{\phi})$. Here $\sigma(\cdot)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

- This model is known as logistic regression – but *note its a model for classification not regression.*

Plots of $p(C_1|\boldsymbol{\phi}) = y(\boldsymbol{\phi}) = \sigma\left(w^T\boldsymbol{\phi}\right)$

- Plots of $p(C_1|f) = \sigma\left(w_1x_1 + w_2x_2\right)$ for a 2D input and different $\boldsymbol{w}$.
- If we threshhold these probabilities at 0.5, we induce a linear decision boundary whose normal is $\boldsymbol{w}$.
- To the right of this have $\sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) > 0$ and to the left $\sigma\left(\boldsymbol{w}^T\boldsymbol{x}\right) < 0$.



Run MatLab function *sigmoidplot2D* from Kevin Murphys' PMTK

Logistic Regression

$$p(C_1|\boldsymbol{\phi}) = \boldsymbol{y}(\boldsymbol{\phi}) = \sigma(\boldsymbol{w}^T\boldsymbol{\phi})$$

- *For an M –dimensional feature space $\boldsymbol{\phi}$, this model has M adjustable parameters.*
- If we had fitted Gaussian class conditional densities using MLE, we would have used
  - $2M$ parameters for the two means and
  - $M(M+1)/2$ parameters for the (shared) covariance.
  - Together with the class prior $p(C_1)$, this gives a total of $M(M+5)/2+1$ parameters, i.e.
  - $\mathcal{O}(M^2)$, in contrast to $\mathcal{O}(M)$ dependence on M in logistic regression.
- For large M, there is advantage in working with logistic regression.
- We now use MLE to determine the parameters in logistic regression. We will use:

$$\frac{d\sigma}{da} = \sigma(1-\sigma)$$

Logistic Regression - Cross-Entropy Error

- For a data set $\{\boldsymbol{\phi}_n, t_n\}$, where $t_n \in \{0,1\}$ *and* $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$, with $n = 1,\ldots,N$, the likelihood function can be written

$$p(\boldsymbol{t}\,|\,\boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n},\ \boldsymbol{t} = (t_1,\ldots,t_N)^T,\ y_n = p(C_1\,|\,\boldsymbol{\phi}_n)$$

- Using this, we define the cross-entropy error function as:

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}\,|\,\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln\left(1-y_n\right)\right\},\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

- Taking the gradient wrt $\boldsymbol{w}$ and using $\dfrac{d\sigma}{da} = \sigma(1-\sigma)$ gives:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n,\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

- The contribution to $\nabla E(\boldsymbol{w})$ from data point n is given by the 'error' $y_n - t_n$ between the target value and the prediction of the model, times the basis $\boldsymbol{\phi}_n$. This is identical to the gradient of the sum-of-squares error function for linear regression.

Logistic Regression - Sequential Update

- We use this result

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} (y_n - t_n)\boldsymbol{\phi}_n,\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$$

for sequential algorithm in which data are presented one at a time.

- The weight vector is updated using

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta\nabla E_n = \boldsymbol{w}^{(\tau)} - \eta(y_n - t_n)\boldsymbol{\phi}_n$$

- Note that for linearly separable data (the hyperplane $\sigma = 0.5, \boldsymbol{w}^T\boldsymbol{\phi} = 0$ separates the two classes), any decision boundary separating the two classes needs to satisfy:

$$\boldsymbol{w}^T\boldsymbol{\phi}_n \begin{cases} \geq 0 \ if\ t_n = 1 \\ < 0 \ otherwise \end{cases}$$

- However, from $\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$ we see that the negative log likelihood is minimized when: $y_n = \sigma\left(\boldsymbol{w}^T\boldsymbol{\phi}_n\right) = t_n, n = 1, \ldots, N$. This requires that the sigmoid function saturates which occurs when its argument

$$\boldsymbol{w}^T\boldsymbol{\phi}_n \to \pm\infty, i.e.\ when \|\boldsymbol{w}\| \to \infty.$$

Logistic Regression- Linearly Separable Data

- MLE can exhibit severe over-fitting for data sets *that are linearly separable.*
- Indeed for linearly separable data, the hyperplane corresponding to $P(C_1|\boldsymbol{x}) = \sigma = 0.5 \rightarrow \boldsymbol{w^T\phi} = 0$ separates the two classes and will have the following:

$$\boldsymbol{w}^T\phi_n = \begin{cases} \geq 0 \; if \; t_n = 1 \\ < 0 \; if \; t_n = 0 \end{cases}$$

- In addition, we can see that the negative log-likelihood is minimized when

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}(y_n - t_n)\boldsymbol{\phi}_n = 0 \Rightarrow y_n = \sigma(\boldsymbol{w}^T\boldsymbol{\phi}_n) = t_n \; \forall n, \; i.e. \; y_n = t_n = 0 \text{ or } y_n = t_n = 1$$

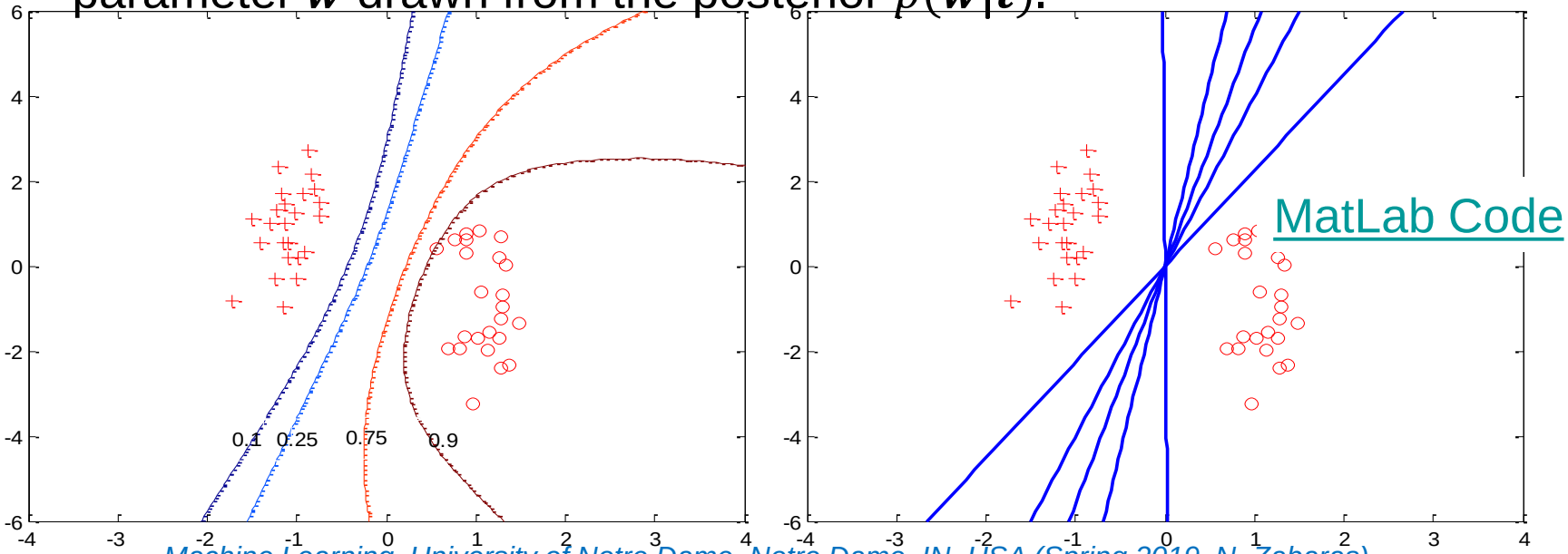
- This occurs when *the sigmoid function is saturated*, which occurs when its argument $\boldsymbol{w}^T\boldsymbol{\phi}$ goes to $\pm\infty$, i.e. *when the magnitude of* $\boldsymbol{w}$ *goes to* ∞.
- The logistic sigmoid function becomes infinitely steep in feature space (step function) so every training point from each class *k is* assigned a posterior probability $p(C_k|\boldsymbol{x}) = 1$.
- There are infinite solutions since any separating hyperplane will give the same posterior probabilities at the training data points.

Logistic Regression Linearly Separable Data

- Maximum likelihood provides no way to favor one such solution over another.
- *Which solution is found depends on the choice of optimization algorithm and on the parameter initialization.*
- The problem arises even if the number of data points is large compared to the number of parameters in the model, so long as the training data set is linearly separable.
- The singularity can be avoided by inclusion of a prior and finding a MAP solution for $\boldsymbol{w}$, or equivalently by adding a regularization term to the error function.

Logistic Regression-Linearly Separable Data

- Bayesian approach to logistic regression for linearly separable data set.
- Left: the predictive distribution using Variational Inference (details will be given in another lecture). The decision boundary lies between the clusters of data points. The contours of the predictive distribution splay out away from the data reflecting the greater uncertainty in the classification of such regions.
- Right: the decision boundaries corresponding to 5 samples of the parameter $\boldsymbol{w}$ drawn from the posterior $p(\boldsymbol{w}|\boldsymbol{t})$.



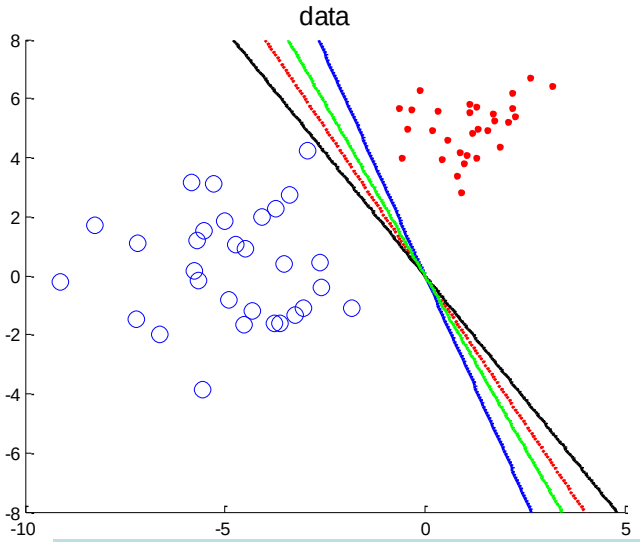
MatLab Code

L_2 Regularization

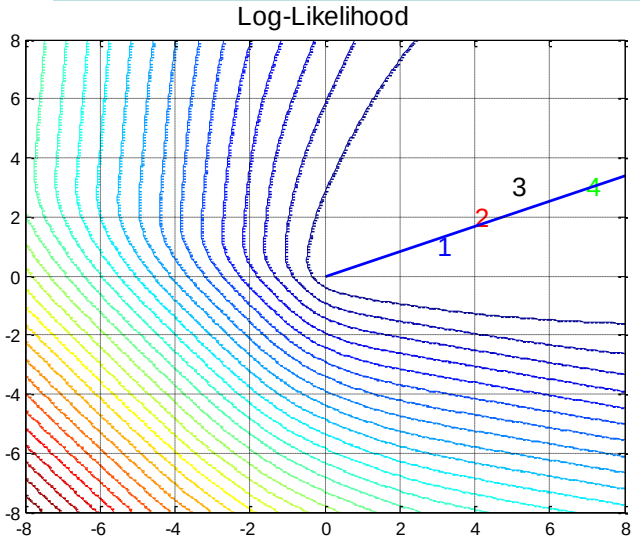
- Regularization is important for classification even with lots of data.
- *For linearly separable data, the MLE is obtained by $||\boldsymbol{w}|| \rightarrow \infty$, corresponding to an infinitely steep sigmoid.* In the two class linearly separable data, we show the Log-likelihood.
- The line is drawn from the origin in the direction of the MLE (which is at ∞). The numbers correspond to the lines on the top Fig.
- This solution is very brittle and does not generalize well.
- To prevent this, we use L_2 regularization.

$$f(\boldsymbol{w}) = NLL(\boldsymbol{w}) + \lambda\, \boldsymbol{w}^T\boldsymbol{w} \text{ (objective function)}$$
$$\boldsymbol{g}(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) + \lambda\, \boldsymbol{w} \text{ (gradient)}$$
$$\boldsymbol{H}(\boldsymbol{w}) = \boldsymbol{H}(\boldsymbol{w}) + \lambda\, \boldsymbol{I} \text{ (Hessian)}$$

- These modifications provide no difficulty with any gradient-based optimization toolbox.



logregLaplaceGirolamiDemo from Kevin Murphys' PMTK



Iterative Reweighted Least Squares

- In linear regression models, the MLE for a Gaussian noise model had a closed-form solution. This was a consequence of the quadratic dependence of the log likelihood function on $\boldsymbol{w}$.
- For logistic regression, there is no closed-form solution due to the nonlinearity of the logistic sigmoid function.
- However, the error function is convex and hence has a unique minimum.
- In addition, it can be minimized by an iterative technique based on *Newton-Raphson using a quadratic approximation to the log likelihood function.*
- The *Newton-Raphson update*, for minimizing a function $E(\boldsymbol{w})$, takes the form

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \boldsymbol{H}^{-1}\nabla E(\boldsymbol{w}), \;\; \boldsymbol{H}_{ij} = \partial^2 E(\boldsymbol{w}) \big/ \partial w_i w_j \;\; (Hessian)$$

IRLS for Linear Regression

- Let us first apply the Newton-Raphson method to the linear regression model with the sum-of-squares error function. The gradient and Hessian of this error function are given by

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\phi(\boldsymbol{x}_n)\right)^2 \Rightarrow \nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}\left(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n) - t_n\right)\boldsymbol{\phi}_n = \boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{\Phi}^T\boldsymbol{t}$$

  where $\boldsymbol{\Phi}$ is the $N \times M$ matrix whose n^{th} row is given by $\boldsymbol{\phi}_n^T$.

- The Newton-Raphson update takes the form:

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{\Phi}^T\boldsymbol{t}\right) = \left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\boldsymbol{t}$$

$$where\ \boldsymbol{H} = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N}\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$$

- This is the standard least-squares solution to *linear regression.*

- *The error function is quadratic and hence the Newton-Raphson gives the exact solution in one step!*

IRLS Applied to Logistic Regression

- We similarly apply the Newton-Raphson to the cross-entropy error function for the logistic regression model:

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t} \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1-t_n) \ln\left(1-y_n\right) \right\},\ y_n = \sigma(a_n),\ a_n = \boldsymbol{w}^T \boldsymbol{\phi}_n \Rightarrow$$

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \left(y_n - t_n\right) \boldsymbol{\phi}_n = \boldsymbol{\Phi}^T \left(\boldsymbol{y} - \boldsymbol{t}\right)$$

$$\boldsymbol{H} = \nabla\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} y_n \left(1-y_n\right) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T = \boldsymbol{\Phi}^T \boldsymbol{R} \boldsymbol{\Phi},\ \boldsymbol{R} = diag\left(y_1\left(1-y_1\right), \ldots, y_n\left(1-y_n\right)\right)$$

  where in the last equation we used $d\sigma / da = \sigma(1-\sigma)$

- The Hessian now depends on $\boldsymbol{w}$ through *the weighting matrix* $\boldsymbol{R}$, since the error function is no longer quadratic.
- Using $0 < y_n < 1$ from the form of the logistic sigmoid function, we see that $\boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} > 0$ for any $\boldsymbol{u}$, and so $\boldsymbol{H}$ is positive definite.
- The error function is a convex function of $\boldsymbol{w}$ and hence has a unique minimum.

IRLS Applied To Logistic Regression

- The Newton-Raphson update then becomes:

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} - \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T (\boldsymbol{y} - \boldsymbol{t}) = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \left\{\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{\Phi}^T (\boldsymbol{y} - \boldsymbol{t})\right\}$$

$$= \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{ where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{t})$$

- The update formula takes the form of a set of normal equations for a weighted least-squares problem.
- *Because $\boldsymbol{R}$ depends on $\boldsymbol{w}$, we apply the normal equations iteratively.*
- For this reason, the algorithm is known as iterative reweighted least squares - IRLS.

$\boldsymbol{w} = \boldsymbol{0}_D$;

$w_0 = \log \bar{t} / (1 - \bar{t})$;

repeat

  $\eta_i = w_0 + \boldsymbol{w}^T \boldsymbol{x}_i$;

  $y_i = sigm(\eta_i)$;

  $r_i = y_i(1 - y_i)$;

  $z_i = \eta_i + (t_i - y_i) / r_i$;

  $\boldsymbol{R} = diag(r_{1:N})$;

  $\boldsymbol{w} = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}$;

until *converged;*

- Rubin, D. B. (1983). Iteratively reweighted least squares. In *Encyclopedia of Statistical Sciences*, Volume 4, pp. 272–275. Wiley.

IRLS Applied To Logistic Regression

$$\boldsymbol{w}^{(new)} = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{t}\right),$$

$$\boldsymbol{R} = diag\left(y_1\left(1-y_1\right), \ldots, y_n\left(1-y_n\right)\right)$$

- The elements of the diagonal weighting matrix $\boldsymbol{R}$ can be interpreted as variances.

- Indeed note that the mean and variance of t in the logistic regression model are given as:

$$\mathbb{E}\left[t\right] = 1 \times \sigma(\boldsymbol{x}) + 0 \times \left(1 - \sigma(\boldsymbol{x})\right) = y$$

$$\text{var}\left[t\right] = \mathbb{E}\left[t^2\right] - \mathbb{E}\left[t\right]^2 = \sigma(\boldsymbol{x}) - \sigma(\boldsymbol{x})^2 = y(1-y)$$

where we used $t^2 = t$ for $t \in \{0, 1\}$.

IRLS Applied To Logistic Regression

$$\boldsymbol{w}^{(new)} = \left(\boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{R}\boldsymbol{z}, \text{where } \boldsymbol{z} = \boldsymbol{\Phi}\boldsymbol{w}^{(old)} - \boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{t}\right),$$

$$\boldsymbol{R} = diag\left(y_1\left(1-y_1\right), \ldots, y_n\left(1-y_n\right)\right)$$

- We can also interpret *IRLS* as the solution to a linearized problem in the space of the variable $a = \boldsymbol{w}^T\boldsymbol{\phi}$.

- The quantity z_n, which corresponds to the n^{th} element of $\boldsymbol{z}$, can be seen as ***an effective target value*** in this space obtained by making a local linear approximation to the logistic sigmoid function around $\boldsymbol{w}^{(old)}$.

$$y_n = \sigma(a_n) = \frac{1}{1+e^{-a_n}}, a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n, \frac{dy_n}{da_n} = y_n\left(1-y_n\right) \Rightarrow$$

$$t_n \cong y_n + \frac{dy_n}{da_n}\Big|_{w^{(old)}}\left(a_n(w) - a_n(w^{(old)})\right)$$

$$a_n(w) \simeq a_n(w^{(old)}) + \frac{da_n}{dy_n}\Big|_{w^{(old)}}(t_n - y_n) = \phi_n^T w^{(old)} - \frac{(y_n - t_n)}{y_n(1-y_n)} = z_n$$

Multiclass Logistic Regression

- For $K > 2$ classes, the posterior probabilities are given by a *softmax transformation of linear functions of the feature variables* as follows:

$$p(C_k \mid \boldsymbol{\phi}) = y_k(\boldsymbol{\phi}) = \frac{e^{a_k}}{\sum_j e^{a_j}}, \quad a_k = \ln\left(p(\boldsymbol{\phi} \mid C_k)\, p(C_k) \right) = \mathbf{w}_k^T \boldsymbol{\phi}$$

- Use maximum likelihood to determine $\{\mathbf{w}_k\}$ for this model.

- We need to compute the derivatives of y_k with respect to all of the activations a_j. They are given as:

$$y_k(\boldsymbol{\phi}) = \frac{e^{a_k}}{\sum_j e^{a_j}} \Rightarrow \frac{\partial y_k}{\partial a_j} = y_k(\delta_{kj} - y_j)$$

- Krishnapuram, B., L. Carin, M. Figueiredo, and A. Hartemink (2005). Learning sparse bayesian classifiers: multi-class formulation, fast algorithms, and generalization bounds. *IEEE Transaction on Pattern Analysis and Machine Intelligence*.

Multiclass Logistic Regression

- Next we write down the likelihood function.
- We use the 1-of-K coding scheme. The target $\boldsymbol{t}_n$ for a feature $\boldsymbol{\phi}_n$ belonging to class C_k is a binary vector with all elements zero except for element k, which equals one.
- The likelihood function is then given by

$$p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \prod_{n=1}^{N} \prod_{k=1}^{K} p(C_k \mid \boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N} \prod_{k=1}^{K} y_{nk}^{t_{nk}},$$

$$y_{nk} = y_k(\boldsymbol{\phi}_n), \boldsymbol{T} \; N \times K \text{ matrix}, T_{nk} = t_{nk}$$

- The negative log likelihood is then

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- This is the cross-entropy error function for the multiclass classification.

Multiclass Logistic Regression

$$E(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- We now take the gradient of the error function with respect to one of the $\boldsymbol{w}_j$.
- Using $y_{nk} = \dfrac{e^{a_{kn}}}{\sum_j e^{a_{jn}}}$, $\log y_{nk} = a_{kn} - \log \sum_j e^{a_{jn}}$, $a_{kn} = \boldsymbol{w}_k^T \boldsymbol{\phi}_n$, $\dfrac{\partial y_{nk}}{\partial a_j} = y_{nk}(\delta_{kj} - y_{nj})$, and $\sum_k t_{nk} = 1$, we obtain

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = \sum_{n=1}^{N} \left(y_{nj} - t_{nj} \right) \boldsymbol{\phi}_n$$

- This is again the form for the gradient derived earlier for the sum-of-squares error function with the linear regression model and the cross-entropy error for logistic regression.

- We *use a sequential algorithm*

$$\boldsymbol{w}_j^{(\tau+1)} = \boldsymbol{w}_j^{(\tau)} - \eta \nabla E_{nj} = \boldsymbol{w}^{(\tau)} - \eta (y_{nj} - t_{nj}) \boldsymbol{\phi}_n$$

General Form: Derivative of log Likelihood

- We have seen that the derivative of the log likelihood for a linear regression model with respect to $\boldsymbol{w}$ for a data point n took the form of the 'error' $y_n - t_n$ times the feature vector $\boldsymbol{\phi}_n$.
- Similarly, for the combination of logistic sigmoid activation function and cross-entropy error function

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t} \mid \boldsymbol{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln y_n + (1 - t_n) \ln \left(1 - y_n\right) \right\},$$

  and for the softmax activation function with the multiclass cross-entropy error function

$$E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\ln p(\boldsymbol{T} \mid \boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

- These are examples of a general result (see here for more discussion)
  - Conditional distribution of the target variable from the exponential family
  - Choosing the activation function as the canonical link function

IRLS Algorithm for the Multiclass Problem

- To find a batch algorithm, we again appeal to the Newton-Raphson update to obtain the corresponding IRLS algorithm for the multiclass problem.

- This requires evaluation of the Hessian matrix that comprises blocks of size $M \times M$ in which block j, k is given by

$$\nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} \left(y_{nj} - t_{nj} \right) \boldsymbol{\phi}_n \Rightarrow \nabla_{\boldsymbol{w}_k} \nabla_{\boldsymbol{w}_j} E(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K) = \sum_{n=1}^{N} y_{nk} \left(\delta_{kj} - y_{nj} \right) \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T$$

  where we used:

$$\nabla_{\boldsymbol{w}_k} y_{nj} = \frac{dy_{nj}}{da_{nk}} \nabla_{\boldsymbol{w}_k} a_{nk} = y_{nk} \left(\delta_{kj} - y_{nj} \right) \boldsymbol{\phi}_n$$

- As with the two-class problem, the Hessian matrix for the multiclass logistic regression model is positive definite and so the error function again has a unique minimum.

- Bishop, C. M. and I. T. Nabney (2008). *Pattern Recognition and Machine Learning: A Matlab Companion*. Springer. In preparation

MAP Estimate

- One can define an appropriate prior

$$p(\boldsymbol{W}) = \prod_k \mathcal{N}(\boldsymbol{w}_k \mid \boldsymbol{0}, \boldsymbol{V}_0)$$

- Using this, the modified negative log-likelihood and its gradient and Hessian are:

$$f'(\boldsymbol{W}) = -\log p(\mathcal{D} \mid \boldsymbol{W}) - \log p(\boldsymbol{W}) = f(\boldsymbol{W}) + \frac{1}{2}\sum_k \boldsymbol{w}_k \boldsymbol{V}_0^{-1} \boldsymbol{w}_k$$

$$\boldsymbol{g}'(\boldsymbol{W}) = \boldsymbol{g}(\boldsymbol{W}) + \boldsymbol{V}_0^{-1} \sum_k^K \boldsymbol{w}_k$$

$$\boldsymbol{H}'(\boldsymbol{W}) = \boldsymbol{H}(\boldsymbol{W}) + \boldsymbol{I}_K \otimes \boldsymbol{V}_0^{-1} \left((KD) \times (KD)\right)$$

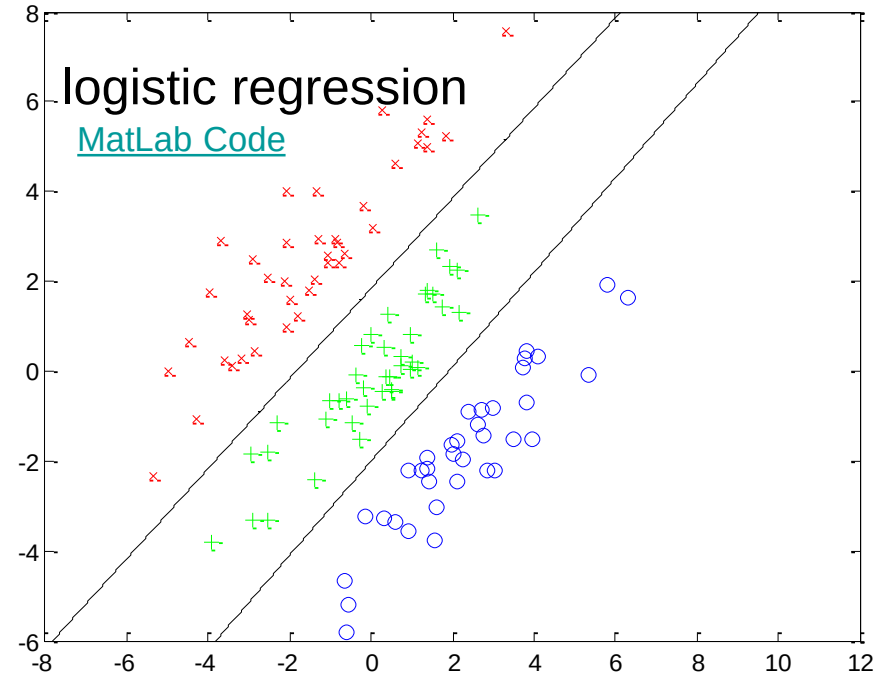
logregFit
from Kevin Murphys' PMTK

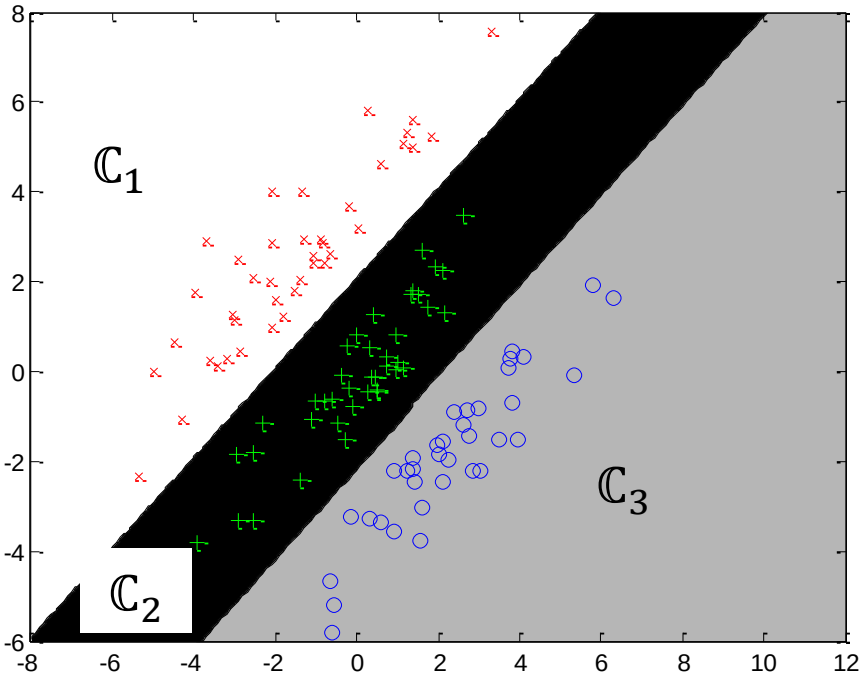
- The symbol $\boldsymbol{I}_C \otimes \boldsymbol{V}_0^{-1}$ is used here to denote Kronecker matrix product.

- One can apply *limited memory BFGS* to find the MAP estimate.

Multiclass Logistic Regression Example

This is the same example of 3 class classification discussed earlier.



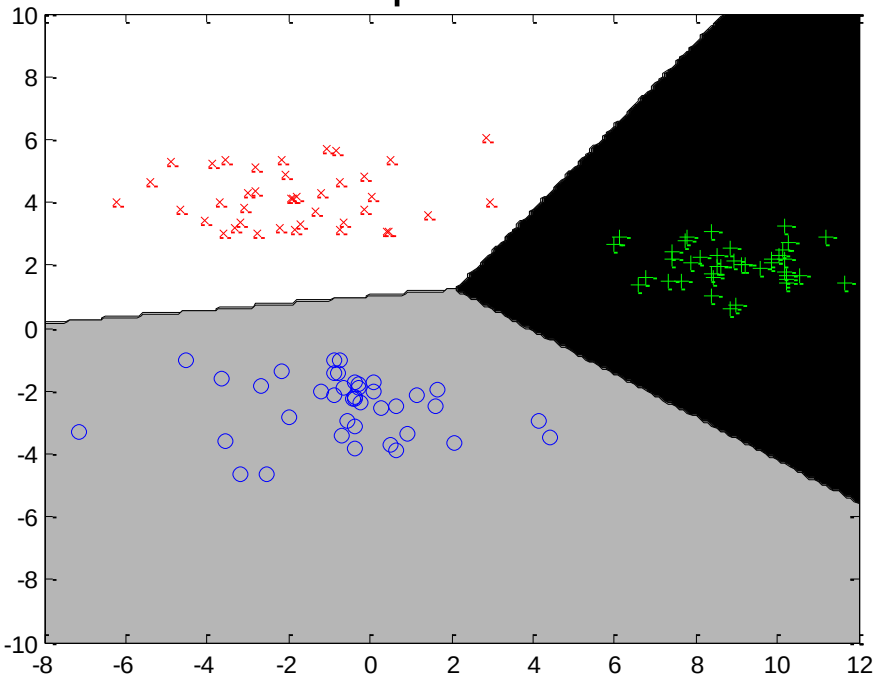


Here, a_1, a_2, a_3 are the feature functions computed by following the logistic regression algorithm. Region C_1, C_2, C_3 in the right plot can be interpreted as:

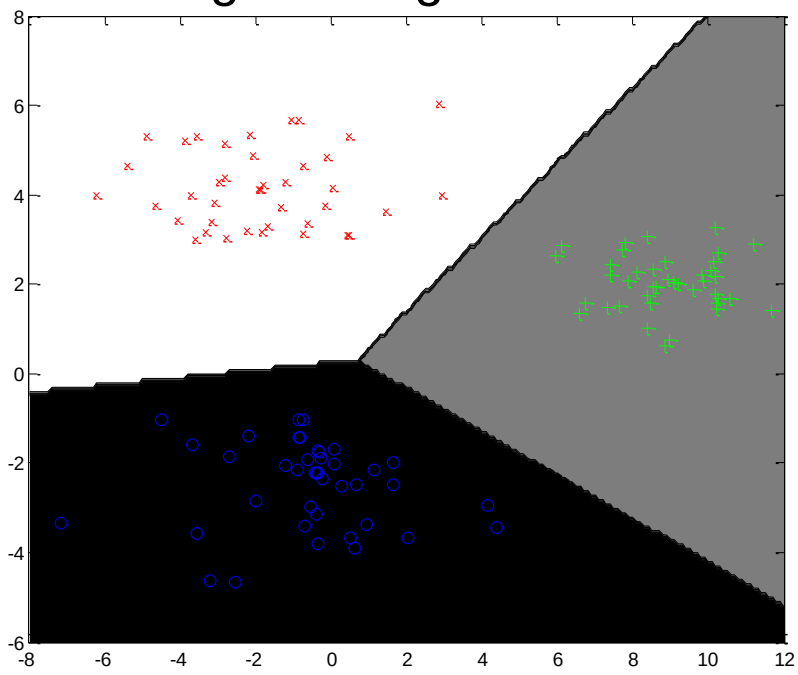
$\mathbb{C}_m: p(C_m|\phi) = \max\left(p(C_1|\phi), p(C_2|\phi), p(C_3|\phi)\right)$ where $p\left(C_k|\phi\right) = \dfrac{\exp\left(a_k\right)}{\sum_j \exp\left(a_j\right)}$

Multiclass Logistic Regression Example

Least-squares



Logistic Regression



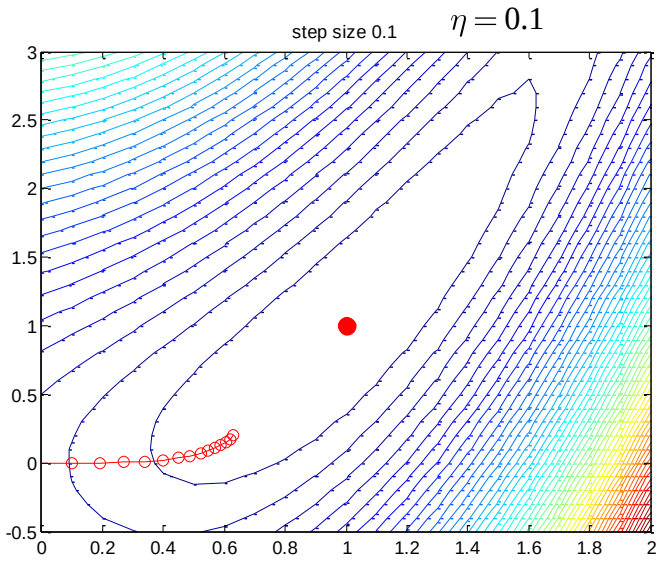
MatLab Code:
Use 'data_3c_2'

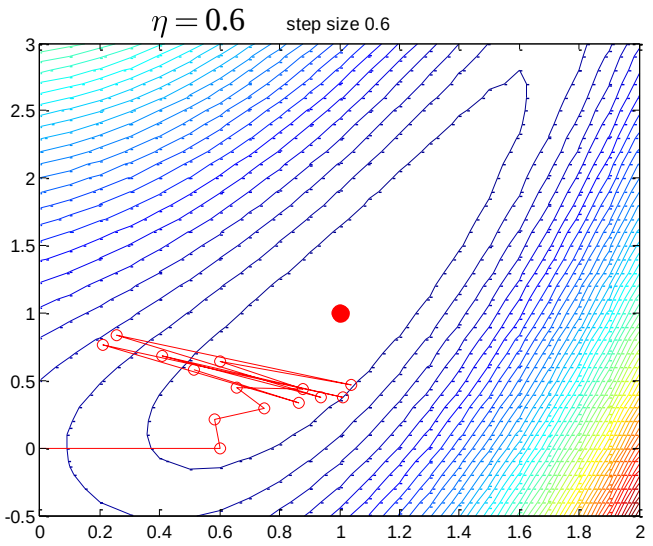
MatLab Code:
Use 'data_3c_2'

Algorithms for Unconstrained Optimization

Steepest Descent Algorithm

- The effect of the learning rate η on the performance of a steepest descent algorithm is shown on a simple function.





$$f(\boldsymbol{\theta}) = 0.5\ \theta_1^2 - \theta_2\ ^2 + 0.5\ \theta_1 - 1\ ^2$$

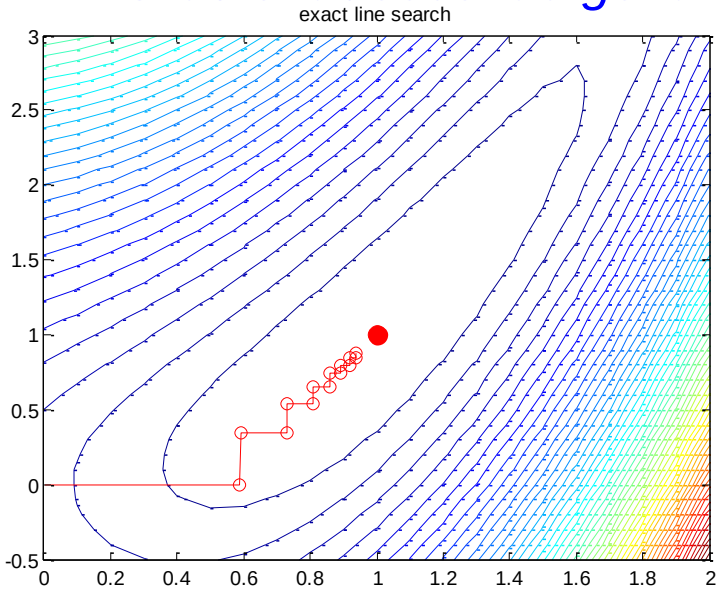
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k = \boldsymbol{\theta}_k - \eta_k \nabla f(\boldsymbol{\theta})\big|_k$$

steepestDescentDemo
from Kevin Murphys' PMTK

- Small step size leads to slow convergence and large step size to never converging. In both cases, the step size is kept constant.

Gradient Descent Algorithm

- *Gradient descent algorithm with line search* is shown below.



$$f(\boldsymbol{\theta}) = 0.5\ \theta_1^2 - \theta_2\ ^2 + 0.5\ \theta_1 - 1\ ^2$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k,\ \boldsymbol{g} = \nabla f$$

$$Pick\ \eta\ to\ minimize:$$

$$\phi(\eta) = f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k$$

$$\approx f\ \boldsymbol{\theta}_k\ + \eta \nabla f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k\ ^T \boldsymbol{d}_k$$

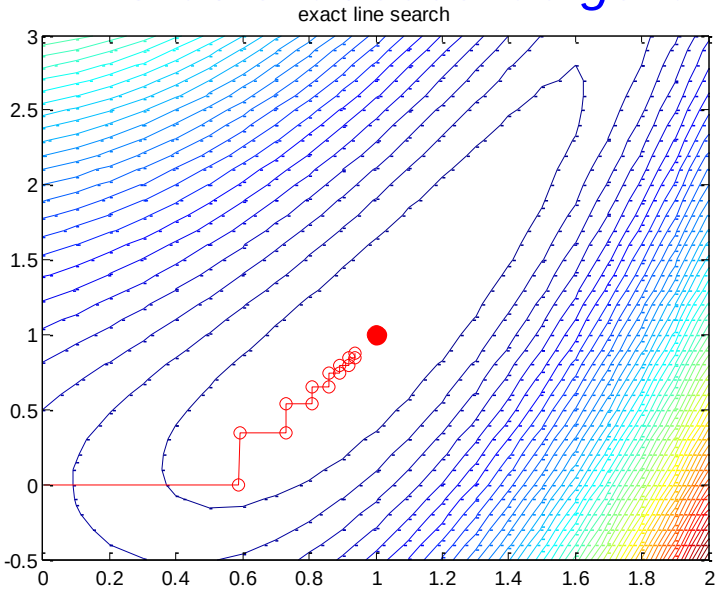
steepestDescentDemo
from Kevin Murphys' PMTK

- Consecutive directions are orthogonal: noticing that $\phi'(\eta) = 0$ leads to: $\boldsymbol{d}^T \boldsymbol{g}_{k+1} = 0,\ \boldsymbol{g}_{k+1} = f'\ \boldsymbol{\theta} + \eta \boldsymbol{d}$.

- So at the end of each step k, either $\nabla f|_{k+1} = 0$ (stationary point was achieved) or $\boldsymbol{d} \perp \nabla f|_{k+1}$ (i.e. the search stops at a point where the gradient is normal to the search direction).

Gradient Descent Example

- *Gradient descent algorithm with line search* is shown below.



[steepestDescentDemo](#)
from [Kevin Murphys' PMTK](#)

$$f(\boldsymbol{\theta}) = 0.5\ \theta_1^2 - \theta_2\ ^2 + 0.5\ \theta_1 - 1\ ^2$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k,\ \boldsymbol{g} = \nabla f$$

$$Pick\ \eta\ to\ minimize:$$

$$\phi(\eta) = f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k$$

$$\approx f\ \boldsymbol{\theta}_k\ + \eta \nabla f\ \boldsymbol{\theta}_k + \eta \boldsymbol{d}_k\ ^T \boldsymbol{d}_k$$

- Bertsekas, D. (1999). [Nonlinear Programming](#) (Second ed.). Athena Scientific.
- Nocedal, J. and S. Wright (2006). [Numerical Optimization](#). Springer.
- Golub, G. and C. F. van Loan (1996). [Matrix computations](#). Johns Hopkins University Press.

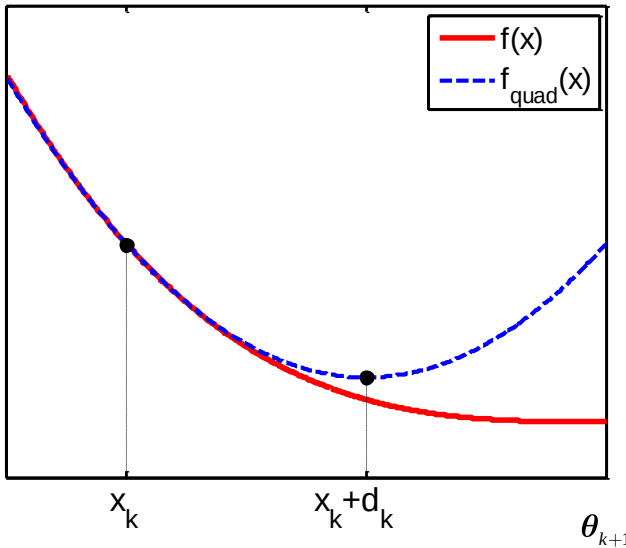
- To avoid the zig-zag path, modify as:

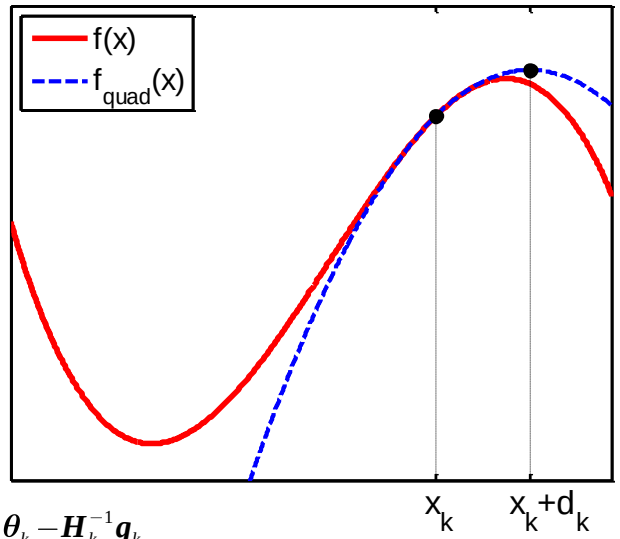
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{g}_k + \mu_k\ \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}\ ,\ \mu_k \in [0,1]$$

- This is the so called *heavy-ball method*.
- Alternatively, one can apply the *conjugate gradient method.*

Newton's Method

- Newton's method takes the Hessian into account: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{H}_k^{-1} \boldsymbol{g}_k, \boldsymbol{g} = \nabla f$





$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k \underbrace{-\boldsymbol{H}_k^{-1}\boldsymbol{g}_k}_{\boldsymbol{d}_k}$

newtonsMethodMinQuad and *newtonsMethodNonConvex* from Kevin Murphys' PMTK

$$f_{quad}(\boldsymbol{\theta}) = f_k + \boldsymbol{g}_k^T(\boldsymbol{\theta}-\boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta}-\boldsymbol{\theta}_k)^T \boldsymbol{H}_k(\boldsymbol{\theta}-\boldsymbol{\theta}_k)$$

$$= \boldsymbol{\theta}^T \boldsymbol{A}\boldsymbol{\theta} + \boldsymbol{b}^T\boldsymbol{\theta} + c$$

$$\boldsymbol{A} = \frac{1}{2}\boldsymbol{H}_k, \boldsymbol{b} = \boldsymbol{g}_k - \boldsymbol{H}_k\boldsymbol{\theta}_k,$$

$$c = f_k - \boldsymbol{g}_k^T\boldsymbol{\theta}_k + \frac{1}{2}\boldsymbol{\theta}_k^T\boldsymbol{H}_k\boldsymbol{\theta}_k$$

- *The step size $d_k = -H_k^{-1} g_k$ is what should be added to θ_k to minimize the 2nd order approximation of f around θ_k.*

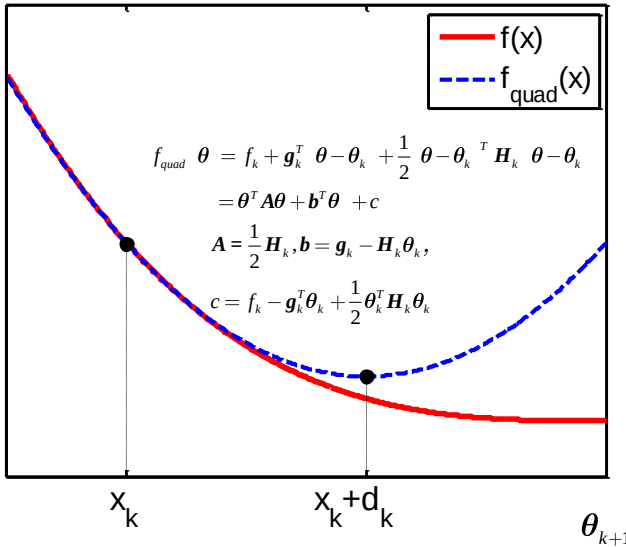
$$f'_{quad}(\boldsymbol{\theta}) = 0 \Rightarrow \boldsymbol{A}\boldsymbol{\theta} = -\frac{1}{2}\boldsymbol{b}$$

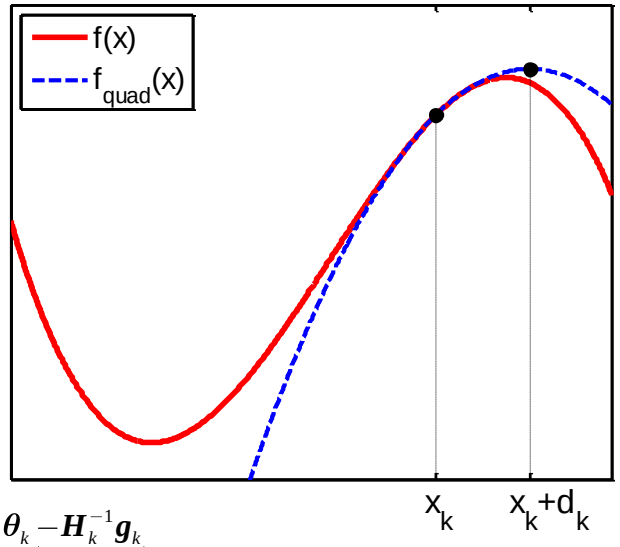
$$\Rightarrow \boldsymbol{H}_k\boldsymbol{\theta}_{k+1} = -(\boldsymbol{g}_k - \boldsymbol{H}_k\boldsymbol{\theta}_k)$$

$$\Rightarrow \boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \boldsymbol{H}_k^{-1}\boldsymbol{g}_k$$

Newton's Method

- Newton's method takes the Hessian into account: $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \boldsymbol{H}_k^{-1} \boldsymbol{g}_k, \boldsymbol{g} = \nabla f$





$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k \underbrace{-\boldsymbol{H}_k^{-1}\boldsymbol{g}_k}_{\boldsymbol{d}_k}$

Initialize $\boldsymbol{\theta}_0$;
for $k = 1, 2, \ldots$ *until convergence* **do**
  Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
  Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
  Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
  Use line search to find stepsize η_k along $\mathbf{d}_k$;
  $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;

- Vandenberghe, L. (2006). Applied numerical computing: Lecture notes.

newtonsMethodMinQuad and newtonsMethodNonConvex from Kevin Murphys' PMTK

- ***H*** *needs to be pos. definite* (holds for convex functions).
  - If not, $\boldsymbol{d}_k$ may not be a descent direction (Fig. on right).
  - For such cases, use the *Levenberg Marquardt method* (adaptively switch between Newton and Steepest descent updates $\boldsymbol{d}_k = -\boldsymbol{g}_k$)

- Also one can use the *truncated Newton method* (solve $\boldsymbol{H}_k \boldsymbol{d}_k = -\boldsymbol{g}_k$ with CG but truncate the iterations when negative curvature is observed).

Limited Memory BFGS

- Computing $\boldsymbol{H}$ is expensive. Quasi Newton methods iteratively build approximations using the gradient vector at each step.

- The BFGS method builds an approximation $\boldsymbol{B}$ (rank–two updates) to the Hessian as (we start with $\boldsymbol{B}_0 = \boldsymbol{I}$):

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{y}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k} - \frac{\boldsymbol{B}_k \boldsymbol{s}_k \quad \boldsymbol{B}_k \boldsymbol{s}_k}{\boldsymbol{s}_k^T \boldsymbol{B}_k \boldsymbol{s}_k}^T$$

$$\boldsymbol{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$$

$$\boldsymbol{y}_k = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$$

- BFGS can be thought of as a diagonal plus low-rank approximation to the Hessian.

- Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer.

Limited Memory BFGS

- Often we provide updates to the inverse of the Hessian:

$$\boldsymbol{C}_{k+1} = \left(\boldsymbol{I} - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right) \boldsymbol{C}_k \left(\boldsymbol{I} - \frac{\boldsymbol{s}_k \boldsymbol{y}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}\right) + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{y}_k^T \boldsymbol{s}_k}, \boldsymbol{C}_k \approx \boldsymbol{H}_k^{-1}$$

$$\boldsymbol{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}$$

$$\boldsymbol{y}_k = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$$

- To avoid storing the Hessian or its inverse, *limited memory BFGS (L-BFGS)* is used where $\boldsymbol{H}_k^{-1}\boldsymbol{g}_k$ is performed by a sequence of inner products with s_k and $\boldsymbol{y}_k$ using only the m most recent (~ 20) pairs of (s_k, y_k).

- This requires only $\mathcal{O}(mD)$ storage vs $\mathcal{O}(D^2)$ for the full Hessian.

- Nocedal, J. and S. Wright (2006). *Numerical Optimization*. Springer.

Bayesian Logistic Regression

Laplace Approximation

- The Laplace approximation is used to find a Gaussian approximation to a probability density.

- We use the fact that the log of a Gaussian is a quadratic function of the M-dimensional variables $\boldsymbol{z}$. Taking a Taylor expansion of $lnf(\boldsymbol{z})$ centered on the mode $\boldsymbol{z}_0$

$$\ln f(\boldsymbol{z}) \approx \ln f(\boldsymbol{z}_0) - \frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)$$

$\boldsymbol{A}$ is the $M \times M$ Hessian matrix (needs to be positive definite)

$$\boldsymbol{A} = -\nabla\nabla \ln f(\boldsymbol{z})|_{\boldsymbol{z}=\boldsymbol{z}_0}$$

- Taking the exponential of this approximation:

$$f(\boldsymbol{z}) \approx f(\boldsymbol{z}_0)\exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)\right\}$$

Laplace Approximation

- We can easily normalize this approximation as follows:

$$q(\boldsymbol{z}) = \frac{|\boldsymbol{A}|^{1/2}}{(2\pi)^{M/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)\right\} = \mathcal{N}\left(\boldsymbol{z} \mid \boldsymbol{z}_0, \boldsymbol{A}^{-1}\right)$$

- To apply the above approximation, $f(\boldsymbol{z})$ doesn't need to be normalized.
- $\boldsymbol{A}$ is the precision matrix that needs to be positive definite, i.e. $\boldsymbol{z}_0$ needs to be a local maximum.
- For multimodal distributions, there are many Laplace approximations depending which mode you are taking the approximation about.
- In Bayesian estimation, the posterior becomes closer to a Gaussian for large number of data (as a result of CLT).

Laplace Approximation

- Often we apply the Laplace approximation to a transformation of the variable. For instance if $0 \leq \tau < \infty$ then we can consider a Laplace approximation of $\ln \tau$.

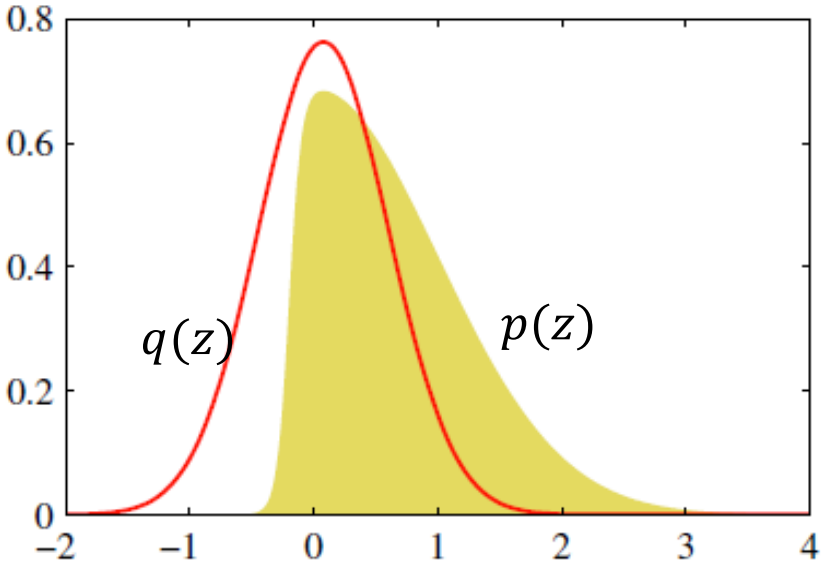
- Since the Laplace approximation is based on a Taylor expansion around the mode, it fails to capture important global properties.

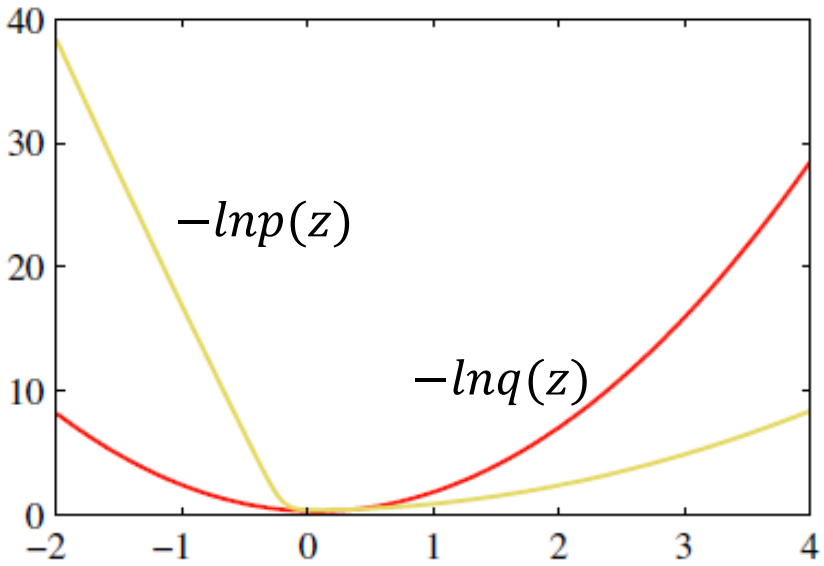
- To capture global properties, alternative approximations are needed.

Laplace Approximation

- The Laplace approximation applied to the distribution $p(z) \propto \exp(-z^2/2)\sigma(20z + 4)$, $\sigma(z)$ being the logistic sigmoid function $\sigma(z) = (1 + e^{-z})^{-1}$.

- The failure to capture global features is clear.





Model Evidence and Model Comparison

- In addition to approximating $p(\boldsymbol{z})$, we can also *obtain an approximation to the normalization constant Z.*

$$Z = \int f(\boldsymbol{z})d\boldsymbol{z} = f(\boldsymbol{z}_0)\int \exp\left\{-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{z}_0)^T \boldsymbol{A}(\boldsymbol{z}-\boldsymbol{z}_0)\right\}d\boldsymbol{z} = f(\boldsymbol{z}_0)\frac{(2\pi)^{M/2}}{|\boldsymbol{A}|^{1/2}}$$

- We can *use this result to obtain an approximation to the model evidence needed in Bayesian model comparison.*

- Consider a data set $\mathcal{D}$ and a set of models $\{\mathcal{M}_i\}$ having parameters $\{\boldsymbol{\theta}_i\}$. For each model we define a likelihood function $p(\mathcal{D}\,|\boldsymbol{\theta}_i, \mathcal{M}_i)$.

- Introduce a prior $p(\boldsymbol{\theta}_i|\mathcal{M}_i)$ over the parameters. We are interested in computing the model evidence $p(\mathcal{D}\,|\mathcal{M}_i)$ for the various models. From now on we omit the conditioning on $\mathcal{M}_i$ to keep the notation uncluttered.

Model Evidence

- The model evidence is:

$$p(\mathcal{D}) = \int \underbrace{p(\mathcal{D}\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta})}_{f(\boldsymbol{\theta})}\,d\boldsymbol{\theta}$$
$$Z$$

- Using our earlier result, we can compute:

$$Z = \int f(\mathbf{z})d\mathbf{z} = f(\mathbf{z}_0)\frac{(2\pi)^{M/2}}{|\boldsymbol{A}|^{1/2}} \Rightarrow$$

$$\ln p(\mathcal{D}) \simeq \underbrace{\ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP})}_{Likelihood} + \underbrace{\ln p(\boldsymbol{\theta}_{MAP}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{A}|}_{Occam\ factor}$$
$$(Penalizes\ Model\ Complexity)$$

- Here $\boldsymbol{\theta}_{MAP}$ is the value of $\boldsymbol{\theta}$ at the mode of the posterior distribution $p(\boldsymbol{\theta}\,|\,\mathcal{D}) \propto p(\mathcal{D}\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta}) = f(\boldsymbol{\theta})$, and $\boldsymbol{A}$ is the Hessian matrix of 2nd derivatives of the negative log posterior.

$$\boldsymbol{A} = -\nabla\nabla \ln p(\mathcal{D}\,|\,\boldsymbol{\theta}_{MAP})\,p(\boldsymbol{\theta}_{MAP}) = -\nabla\nabla \ln p(\boldsymbol{\theta}_{MAP}\,|\,\mathcal{D})$$

Bayesian Information Criterion (BIC)

- Assuming that the Gaussian prior distribution over parameters is broad and that the Hessian has full rank, we can approximate the model evidence very roughly using

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) + \ln p(\boldsymbol{\theta}_{MAP}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{A}|$$

$$\simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) - \frac{1}{2}M\ln N$$

- N is the number of data points, M is the number of parameters in $\boldsymbol{\theta}$ and we have omitted additive constants. This is the Bayesian Information Criterion (BIC). The proof is highlighted in the following slides.

- Compared to AIC (*Akaike information criterion)*

$$\ln p(\mathcal{D}|\widehat{\boldsymbol{\theta}}_{MLE}) - M$$

BIC penalizes model complexity more heavily.

- Akaike, H. (1974). A new look at statistical model identification. *IEEE Trans. on Automatic Control* **19**, 716–723.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics* **6**, 461–464.

Derivation of the BIC Criterion

- The BIC approximation can be viewed as a *large N approximation to the log model evidence*. We have:

$$\boldsymbol{A} = -\nabla\nabla \ln\left(p(|\boldsymbol{\theta}_{MAP})\, p(\boldsymbol{\theta}_{MAP})\right) = \boldsymbol{H} - \nabla\nabla \ln p(\boldsymbol{\theta}_{MAP}),\ \ \boldsymbol{H} = -\nabla\nabla \ln p(\mathcal{D}\,|\,\boldsymbol{\theta}_{MAP})$$

- If $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}\,|\,\boldsymbol{m}, \boldsymbol{V}_0)$, the above equation is simplified:

$$\boldsymbol{A} = \boldsymbol{H} + \boldsymbol{V}_0^{-1}$$

from which we see that

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) + \ln p(\boldsymbol{\theta}_{MAP}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{A}| \Rightarrow$$

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) - \frac{1}{2}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m})^T \boldsymbol{V}_0^{-1}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m}) - \frac{1}{2}\ln|\boldsymbol{H}| + const$$

Derivation of the BIC Criterion

- Since we assume i.i.d. data, $\boldsymbol{H} = -\nabla\nabla \ln p(\mathcal{D} \mid \boldsymbol{\theta}_{MAP})$ consists of a sum of terms, and we can consider the following approximation:

$$\boldsymbol{H} = \sum_{n=1}^{N} \boldsymbol{H_n} = N\widehat{\boldsymbol{H}},\ \widehat{\boldsymbol{H}} = \frac{\sum_{n=1}^{N} \boldsymbol{H}_n}{N}$$

- Combining this with the properties of the determinant, we have

$$ln|\boldsymbol{H}| = ln|N\widehat{\boldsymbol{H}}| = ln\left(N^M|\widehat{\boldsymbol{H}}|\right) = M\ln N + ln\left(|\widehat{\boldsymbol{H}}|\right)$$

where M is the dimensionality of $\boldsymbol{\theta}$. Note that we are assuming that $\widehat{\boldsymbol{H}}$ has full rank M.

- We can now substitute the above equation in:

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) - \frac{1}{2}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m})^T \boldsymbol{V}_0^{-1}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m}) - \frac{1}{2}\ln|\boldsymbol{H}| + const$$

Derivation of the BIC Criterion

$$\ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) - \frac{1}{2}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m})^T V_0^{-1}(\boldsymbol{\theta}_{MAP} - \boldsymbol{m}) - \frac{1}{2} M \ln N - \frac{1}{2} ln\left(|\widehat{\boldsymbol{H}}|\right) + const$$

- We finally obtain the required result by dropping the $ln\left(|\widehat{\boldsymbol{H}}|\right)$ term since its $\mathcal{O}(1)$ compared to lnN and dropping the 2nd term with the assumption of prior with broad variance.

- Thus finally we have:

$$BIC: \quad \ln p(\mathcal{D}) \simeq \ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP}) - \frac{1}{2} M \ln N + const$$

AIC and BIC

- Complexity measures such as AIC and BIC have the virtue of being easy to evaluate, but can also give misleading results.

- In particular, the assumption that the Hessian matrix has full rank is often not valid since many of the parameters are not 'well-determined'.

- We can use

$$\ln p(\mathcal{D}) \simeq \underbrace{\ln p(\mathcal{D}|\boldsymbol{\theta}_{MAP})}_{Likelihood} + \underbrace{\ln p(\boldsymbol{\theta}_{MAP}) + \frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln|\boldsymbol{A}|}_{\substack{Occam\ Factor \\ Penalizes\ Model\ Complexity}}$$

to obtain a more accurate estimate of the model evidence starting from the Laplace approximation.

Bayesian Logistic Regression

- Exact Bayesian inference for logistic regression is intractable.
- Evaluation of the posterior distribution would require normalization of the product of a prior distribution and a likelihood function that is a product of logistic sigmoid functions one for every data point.
- Evaluation of the predictive distribution is similarly intractable.
- Here we consider the application of the Laplace approximation for the posterior. This requires evaluation of the 2nd derivatives of the log posterior (Hessian).

- Spiegelhalter, D. and S. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks* **20**, 579–605.
- MacKay, D. J. C. (1992b). The evidence framework applied to classification networks. *Neural Computation* **4**(5), 720–736.
- Kuss and C. Rasmussen (2005). Assessing approximate inference for binary gaussian process classification. *J. of Machine Learning Research 6*, 1679–1704.

Bayesian Logistic Regression

- Because we seek a Gaussian representation for the posterior, we consider a Gaussian prior:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_0, \boldsymbol{S}_0)$$

where $\boldsymbol{m}_0, \boldsymbol{S}_0$ are fixed hyperparameters.

- The posterior distribution over $\boldsymbol{w}$ is given as:

$$p(\boldsymbol{w} \mid \boldsymbol{t}) \propto p(\boldsymbol{w})\, p(\boldsymbol{t} \mid \boldsymbol{w})$$

where $\boldsymbol{t} = (t_1, \ldots, t_N)^T$. We take the log of both sides, and substitute for the prior distribution and for the likelihood function

$$p(\boldsymbol{t} \mid \boldsymbol{w}) = \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n}$$

,

Bayesian Logistic Regression

$$\ln p(\boldsymbol{w}\,|\,\boldsymbol{t}) \propto -\frac{1}{2}(\boldsymbol{w}-\boldsymbol{m}_0)^T \boldsymbol{S}_0^{-1}(\boldsymbol{w}-\boldsymbol{m}_0)$$

$$+\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + const.,$$

$$where\ y_n = \sigma\left(\boldsymbol{w}^T\boldsymbol{\phi}_n\right)$$

- To obtain a Gaussian approximation to the posterior distribution, first maximize the posterior distribution to give the MAP solution $\boldsymbol{w_{MAP}}$, thus defining the mean of the Gaussian.
- The covariance is then given by the inverse of the matrix of 2nd derivatives of the negative log likelihood:

$$\boldsymbol{S}_N^{-1} = -\nabla\nabla \ln p(\boldsymbol{w}\,|\,\boldsymbol{t}) = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n\boldsymbol{\phi}_n^T$$

Gaussian Approximation to the Posterior

- The Gaussian approximation to the posterior now takes the form:

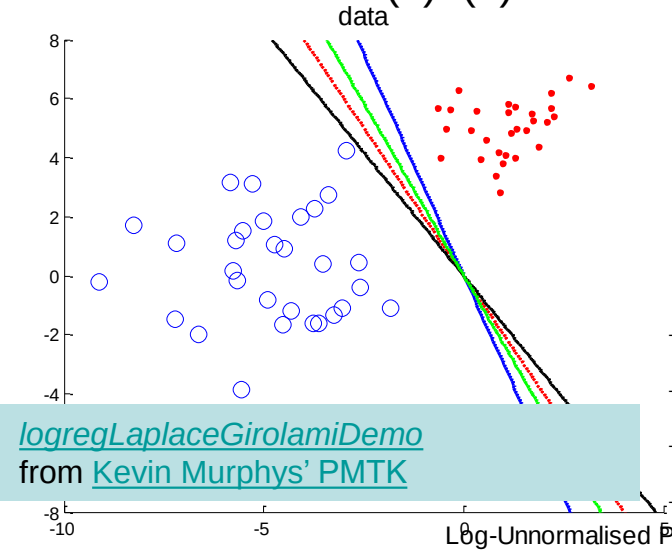
$$q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{m}_{MAP}, \boldsymbol{S}_N)$$

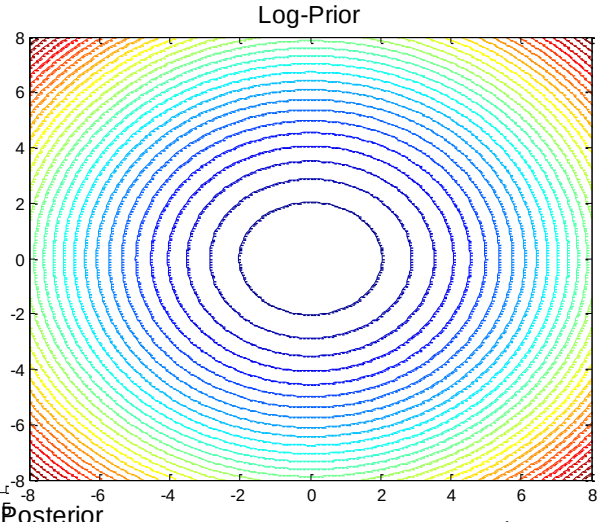
$$\boldsymbol{S}_N^{-1} = -\nabla\nabla \ln p(\boldsymbol{w} \mid \boldsymbol{t}) = \boldsymbol{S}_0^{-1} + \sum_{n=1}^{N} y_n(1-y_n)\boldsymbol{\phi}_n \boldsymbol{\phi}_n^T$$

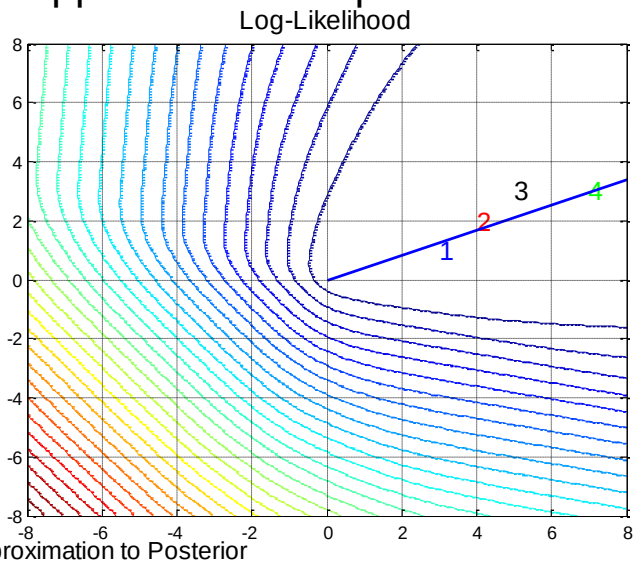
- To make predictions we need to be able to marginalize with respect to this distribution. That's not an easy task.

Gaussian Approximation to the Posterior

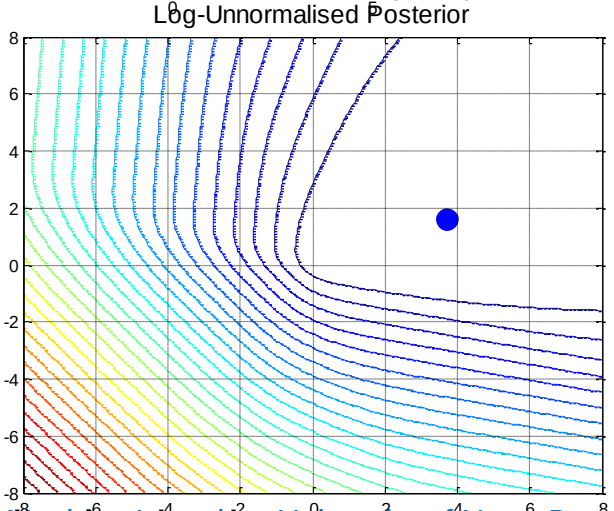
- (a) Two-class data in 2d. (b) Log Prior $\mathcal{N}(\boldsymbol{w}|0,100\boldsymbol{I})$ (c) Log-likelihood for a logistic regression model. The line shown is drawn from the origin in the direction of the MLE (which is at ∞). The numbers correspond to the 4 points in $\boldsymbol{w}$-space corresponding to the lines in (a). (d) Unnormalized log posterior. (d) Laplace approximation to posterior.

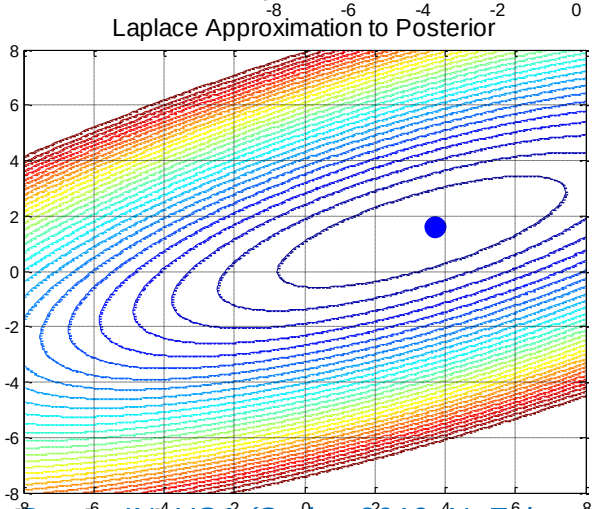






logregLaplaceGirolamiDemo
from Kevin Murphys' PMTK





Predictive Distribution: Plug-In Approximation

- The posterior predictive distribution has the form

$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \int p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{t}) d\boldsymbol{w}$$

- The simplest approximation is the plug-in approximation (using the posterior mean) which for binary classification takes the form:

$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \int p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{t}) d\boldsymbol{w} \approx p\left(C_1 \mid \boldsymbol{\phi}, \mathbb{E}\left[\boldsymbol{w}\right]\right)$$

- This approximation underestimates the uncertainty and better approximations are needed.

Predictive Distribution: MC Approximation

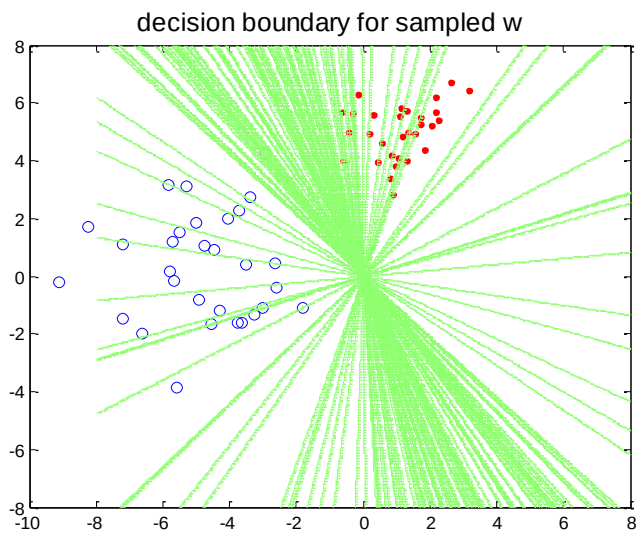
- By taking samples from the posterior $\boldsymbol{w}^s \sim p(\boldsymbol{w} \mid \boldsymbol{t})$

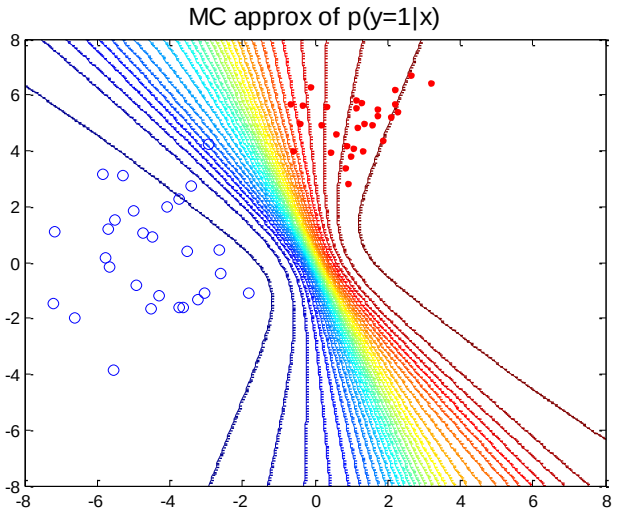
$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \int p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{t}) d\boldsymbol{w} \approx \frac{1}{S} \sum_{s=1}^{S} \sigma\left(\boldsymbol{w}^{sT} \boldsymbol{\phi}(\boldsymbol{x})\right)$$

- This technique can be extended to the multi-class case.
- If we have approximated the posterior using MC, we can reuse these samples for prediction.
- If we made a Gaussian (Laplace) approximation to the posterior, we can draw *independent* samples from the Gaussian using standard methods.

Predictive Distribution: MC Approximation

- By averaging over multiple predictions, we see that the uncertainty in the decision boundary "splays out" as we move away from the training data.
- So although the decision boundary is linear, the posterior predictive density is not linear.
- Note also that the posterior mean decision boundary is roughly equally far from both classes.



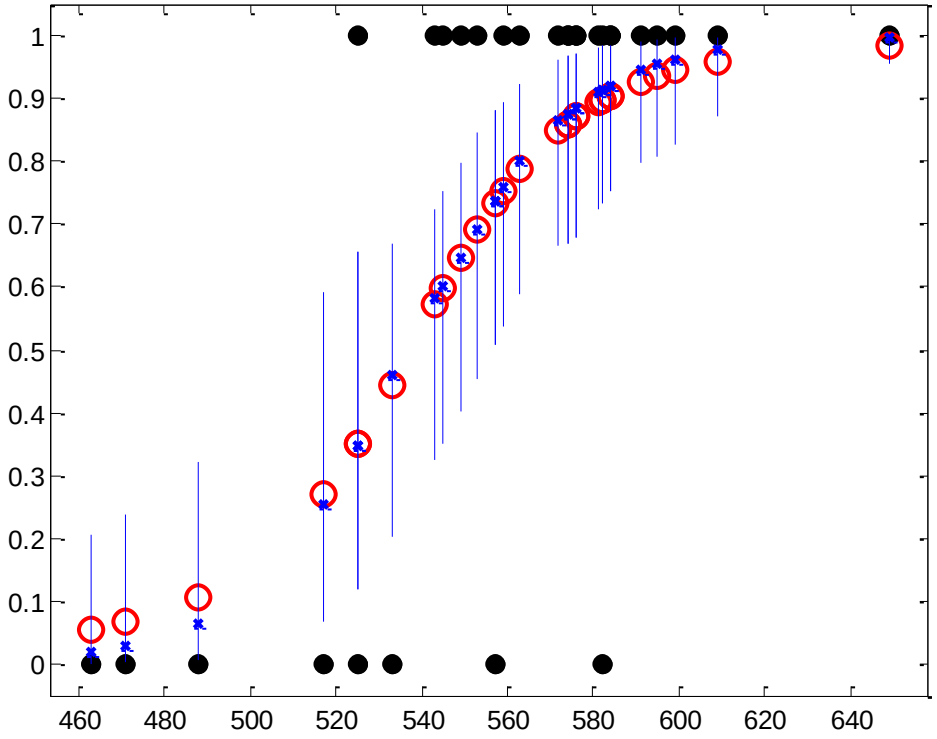


logregLaplaceGirolamiDemo
from Kevin Murphys' PMTK

Predictive Distribution: MC Approximation

- *Red dots:* the mean of the posterior predictive evaluated at the training data. *Vertical blue lines*: 95% credible intervals for the posterior predictive; *Small blue star:* the median.

- With the Bayesian approach, we are able to model our uncertainty about the probability a student will pass the exam based on his SAT score.



logregSATdemoBayes
from Kevin Murphys' PMTK

Predictive Distribution: Probit Approximation

- The predictive distribution for class C_1, given a new feature vector $\boldsymbol{\phi}(\boldsymbol{x})$, *is* obtained by marginalizing with respect to the *posterior distribution $p(\boldsymbol{w}|\boldsymbol{t})$, which is itself approximated by a Gaussian distribution $q(\boldsymbol{w})$ so*

$$p(C_1|\boldsymbol{\phi},\boldsymbol{t}) = \int p(C_1|\boldsymbol{\phi},\boldsymbol{w})\, p(\boldsymbol{w}|\boldsymbol{t}) d\boldsymbol{w} \simeq \int \sigma(\boldsymbol{w}^T\boldsymbol{\phi})\, q(\boldsymbol{w}) d\boldsymbol{w}$$

  The corresponding probability for class C_2 is given simply by $p(C_2|\boldsymbol{\phi},\boldsymbol{t}) = 1 - p(C_1|\boldsymbol{\phi},\boldsymbol{t})$.

- The function $\sigma(\boldsymbol{w}^T\boldsymbol{\phi})$ depends on $\boldsymbol{w}$ only through its projection onto $\boldsymbol{\phi}$. Denoting $a = \boldsymbol{w}^T\boldsymbol{\phi}$, we have

$$\sigma(\boldsymbol{w}^T\boldsymbol{\phi}) = \int \delta(a - \boldsymbol{w}^T\boldsymbol{\phi})\sigma(a) da \Rightarrow \int \sigma(\boldsymbol{w}^T\boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w} = \int \sigma(a) \underbrace{p(a)}_{\int \delta(a-\boldsymbol{w}^T\boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w}} da$$

Predictive Distribution: Probit Approximation

- We can evaluate $p(a) = \int \delta(a - \boldsymbol{w}^T \boldsymbol{\phi}) q(\boldsymbol{w}) d\boldsymbol{w}$ by noting that the delta function imposes a linear constraint on $\boldsymbol{w}$ and so forms a marginal distribution from the joint distribution $q(\boldsymbol{w})$ by integrating out all directions orthogonal to $\boldsymbol{\phi}$.
- Because $q(\boldsymbol{w})$ is Gaussian, the marginal distribution will also be Gaussian.
- We evaluate the mean and variance of this distribution by taking moments, and interchanging the order of integration over a and $\boldsymbol{w}$, so that

$$\mu_a = \mathbb{E}[a] = \int p(a) a \, da = \int_{q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} | \boldsymbol{w}_{MAP}, \boldsymbol{S}_N)} q(\boldsymbol{w}) \, \boldsymbol{w}^T \boldsymbol{\phi} \, d\boldsymbol{w} = \boldsymbol{w}_{MAP}^T \boldsymbol{\phi}$$

$$\sigma_a^2 = \mathrm{var}[a] = \int p(a) \left\{ a^2 - \mathbb{E}[a^2] \right\} da$$

$$= \int_{q(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} | \boldsymbol{w}_{MAP}, \boldsymbol{S}_N)} q(\boldsymbol{w}) \left\{ \left(\boldsymbol{w}^T \boldsymbol{\phi} \right)^2 - \left(\boldsymbol{w}_{MAP}^T \boldsymbol{\phi} \right)^2 \right\} d\boldsymbol{w} = \boldsymbol{\phi}^T \boldsymbol{S}_N \boldsymbol{\phi}$$

Predictive Distribution: Probit Approximation

- Note that the distribution $p(a) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{w}_{MAP}^T \boldsymbol{\phi}, \boldsymbol{\phi}^T \boldsymbol{S}_N \boldsymbol{\phi})$ takes the same form as the predictive distribution for the linear regression model with the noise variance set to zero.
- Thus our variational approximation to the predictive distribution is:

$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \int \sigma(a) p(a) da = \int_{\Phi(\lambda a)} \sigma(a) \mathcal{N}(a \mid \mu_a, \sigma_a^2) da$$

- We *approximate the logistic sigmoid function $\sigma(a)$ with the inverse probit function $\Phi(a)$.*
- To obtain the best approximation to the logistic function we rescale the horizontal axis, so that we *approximate σ(a) by $\Phi(\lambda a)$. We find λ by requiring that the two functions have the same slope at the origin, which gives $\lambda^2 = \pi/8$.*

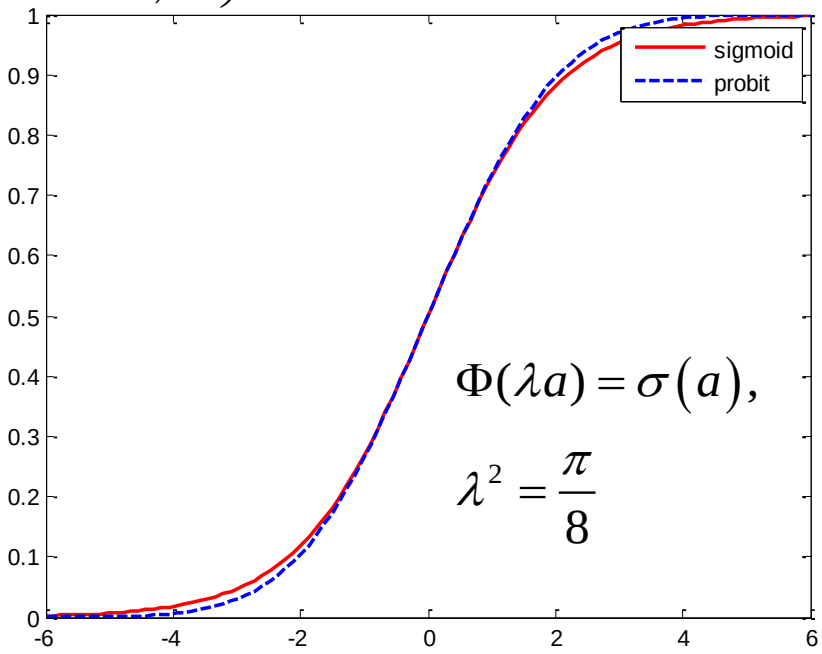
- Spiegelhalter, D. and S. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks* **20**, 579–605.
- MacKay, D. (1992b). The evidence framework applied to classification networks. *Neural Comp.* **4**(5), 720–736.
- Barber, D. and C. M. Bishop (1998a). Ensemble learning for multi-layer networks. In M. I. Jordan, K. J. Kearns, and S. A. Solla (Eds.), *Advances in Neural Information Processing Systems*, Volume 10, pp. 395–401.

Predictive Distribution: Probit Approximation

- The convolution below can be performed analytically:

$$\int \Phi(\lambda a)\mathcal{N}(a \mid \mu, \sigma^2)da = \Phi\left(\frac{\mu}{\left(\lambda^{-2}+\sigma^2\right)^{1/2}}\right)$$

where $\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta \mid 0,1)d\theta$



- We now apply the approximation $\sigma(a) \simeq \Phi(\lambda a)$ to the inverse probit functions appearing on both sides of this equation, leading to the following approximation for the convolution of a logistic sigmoid with a Gaussian

$$\int \sigma(a)\mathcal{N}(a \mid \mu, \sigma^2)da = \sigma\left(\kappa\left(\sigma^2\right)\mu\right), \kappa\left(\sigma^2\right) = \left(1+\frac{\pi\sigma^2}{8}\right)^{-1/2}$$

probitPlot
from Kevin Murphys' PMTK

Appendix: Convolution of Φ(λa) with a Gaussian

- We will highlight below the proof of

$$\int_{-\infty}^{\infty}\Phi(\lambda a)\mathcal{N}(a\,|\,\mu,\sigma^2)da=\Phi\left(\frac{\mu}{\left(\lambda^{-2}+\sigma^2\right)^{1/2}}\right),\ \text{where}\ \Phi(b)=\int_{-\infty}^{b}\mathcal{N}(\theta\,|\,0,1)d\theta$$

- The derivative of the rhs wrt μ is: $\mathcal{N}\left(\frac{\mu}{\left(\lambda^{-2}+\sigma^2\right)^{1/2}}\,|\,0,1\right)\frac{1}{\left(\lambda^{-2}+\sigma^2\right)^{1/2}}$

- We use $a=\mu+\sigma z$ and write the derivative wrt μ on the lhs:

$$\int_{-\infty}^{\infty}\frac{d}{d\mu}\left\{\Phi\left(\lambda(\mu+\sigma z)\right)\mathcal{N}\left((\mu+\sigma z)\,|\,\mu,\sigma^2\right)\right\}\sigma dz=\int_{-\infty}^{\infty}\left\{\frac{d\Phi\left(\lambda(\mu+\sigma z)\right)}{d\mu}\mathcal{N}\left((\mu+\sigma z)\,|\,\mu,\sigma^2\right)+\Phi\left(\lambda(\mu+\sigma z)\right)\frac{d\mathcal{N}\left((\mu+\sigma z)\,|\,\mu,\sigma^2\right)}{d\mu}\right\}\sigma dz=$$

$$\int_{-\infty}^{\infty}\left\{\mathcal{N}\left(\lambda(\mu+\sigma z)\,|\,0,1\right)\lambda\mathcal{N}\left((\mu+\sigma z)\,|\,\mu,\sigma^2\right)\right\}\sigma dz=\frac{1}{2\pi}\int_{-\infty}^{\infty}e^{-\frac{(\lambda(\mu+\sigma z))^2}{2}}e^{-\frac{z^2}{2}}\lambda dz\underset{M=\mu/\sqrt{\lambda^{-2}+\sigma^2}}{=}\frac{1}{\sqrt{2\pi}}e^{-M^2/2}\lambda\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}e^{-\frac{\left(\sqrt{\lambda^2\sigma^2+1}z+\lambda\sigma M\right)^2}{2}}dz=$$

$$\frac{1}{\sqrt{2\pi}}e^{-M^2/2}\lambda\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}e^{-\frac{\chi^2}{2}}\frac{1}{\sqrt{\lambda^2\sigma^2+1}}d\chi=\frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{\sigma^2+\lambda^{-2}}}e^{-M^2/2}$$

- The derivatives wrt μ match. Thus the following must hold:

$$\int_{-\infty}^{\infty}\Phi(\lambda a)\mathcal{N}(a\,|\,\mu,\sigma^2)da=\Phi\left(\frac{\mu}{\left(\lambda^{-2}+\sigma^2\right)^{1/2}}\right)+const\left(\sigma,\lambda\right)$$

- Take $\lambda=0$ to see that the constant of integration is zero.

Predictive Distribution: Probit Approximation

$$\int \sigma(a)\mathcal{N}(a \mid \mu, \sigma^2)da = \sigma\left(\kappa\left(\sigma^2\right)\mu\right), \qquad \kappa\left(\sigma^2\right) = \left(1 + \frac{\pi\sigma^2}{8}\right)^{-1/2}$$

- Applying this result to

$$p(C_1 \mid \boldsymbol{t}) = \int \sigma(a)p(a)da = \int_{\Phi(\lambda a)} \sigma(a)\mathcal{N}(a \mid \mu_a, \sigma_a^2)da$$

finally gives:

$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \sigma\left(\kappa\left(\sigma_a^2\right)\mu_a\right)$$

where $\mu_a = \boldsymbol{w}_{MAP}^T \boldsymbol{\phi}$ and $\sigma_a^2 = \boldsymbol{\phi}^T \boldsymbol{S}_N \boldsymbol{\phi}$

Predictive Distribution: Probit Approximation

$$p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = \sigma\left(\kappa\left(\sigma_a^2\right)\mu_a\right)$$

- Note that the decision boundary corresponding to $p(C_1 \mid \boldsymbol{\phi}, \boldsymbol{t}) = 0.5$ is given by $\mu_a = \boldsymbol{w}_{MAP}^T \boldsymbol{\phi} = 0$, *which is the same as the decision boundary obtained by using the MAP value for* $\boldsymbol{w}$*.*

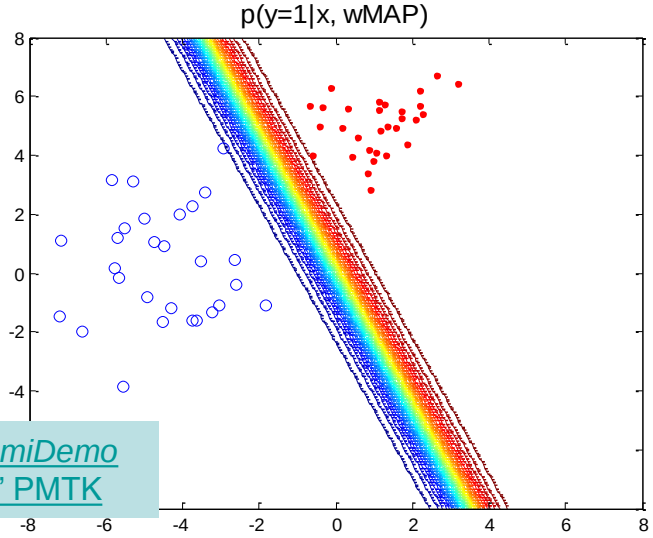
- Thus for a decision criterion based on minimizing the misclassification rate and with equal prior probabilities, the marginalization over $\boldsymbol{w}$ has no effect.

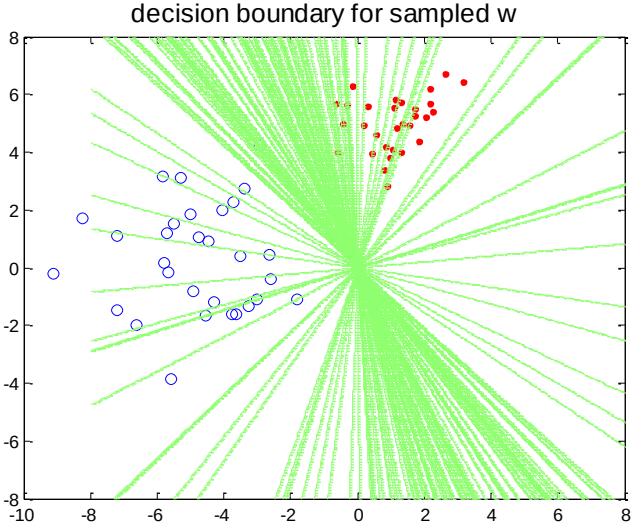
- For more complex decision criteria, using the predictive distribution is essential.

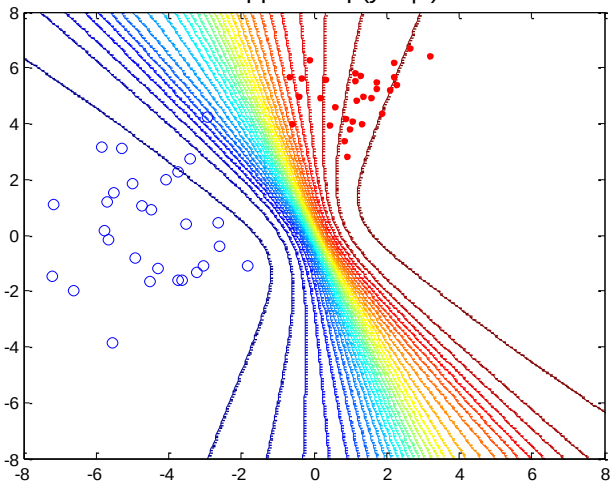
- Rasmussen, C. E. and C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

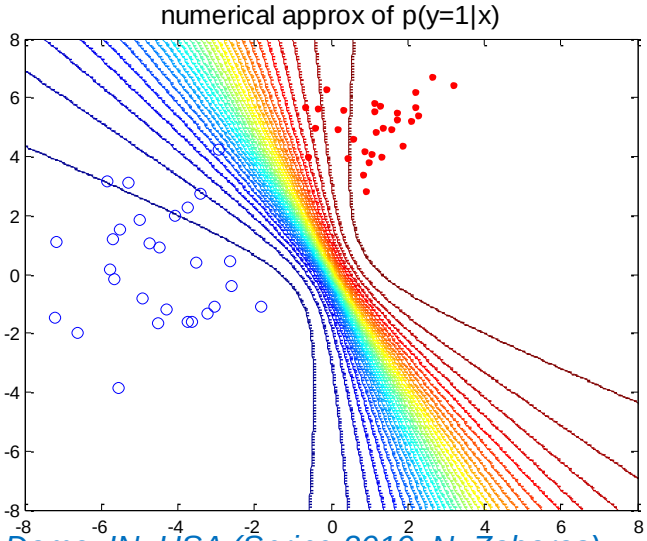
Posterior Predictive

- Posterior predictive distribution for a logistic regression model in 2d. (a) contours of $p(y = 1|\boldsymbol{x}, \boldsymbol{w}_{map})$. (b) samples from the posterior predictive distribution. (c) Averaging over these samples. (d) probit approximation.









logregLaplaceGirolamiDemo from Kevin Murphys' PMTK

Outlier Detection

- It is useful to detect outliers in the data. In e.g. regression, this can be performed by computing

$$r_i = y_i - \hat{y}_i, \hat{y}_i = \hat{\boldsymbol{w}}^T \boldsymbol{x}_i$$

- These values should follow a $\mathcal{N}(0, \sigma^2)$ distribution.
- This is assessed by a qq-plot, where we plot the N theoretical quantiles of a Gaussian distribution against the N empirical quantiles of the r_i.
- Points that deviate from the straight line are potential outliers.
- Methods based on residuals do not work for binary data since they rely on asymptotic statistics.

Outlier Detection in Logistic Regression

- In logistic regression, we define outliers as points which have low probability.

- Outliers are defined as points with low probability under the cross-validated posterior predictive distribution,

$$p(t_i \mid \boldsymbol{\phi}_i, \boldsymbol{\phi}_{-i}, \boldsymbol{t}_{-i}) = \int p\left(t_i \mid \boldsymbol{\phi}_i, \boldsymbol{w}\right) \underbrace{\prod_{i' \neq i} p\left(t_i^{'} \mid \boldsymbol{\phi}_i^{'}, \boldsymbol{w}\right) p(\boldsymbol{w})}_{\text{Posterior without point i in likelihood}} \, d\boldsymbol{w}$$

- This can be computed by sampling.

- Gelfand, A. (1996). Model determination using sampling-based methods. In Gilks, Richardson, and Spiegelhalter (Eds.), *Markov Chain Monte Carlo in Practice*. Chapman & Hall.
- Johnson, V. and J. Albert (1999). *Ordinal data modeling*. Springer.

Probit Regression

Probit Regression

- We have seen that for the exponential family of class-conditional distributions, the resulting posterior class probabilities are given by a logistic (or softmax) transformation acting on a linear function of the feature variables.
- Not all choices of class-conditional densities (e.g. Gaussian mixtures) give rise to this form for the posterior probabilities.
- We need to explore other types of discriminative probabilistic models.
- We return to the two-class case within the framework of generalized linear models so that

$$p(t = 1|a) = f(a)$$

where $a = \boldsymbol{w}^T\boldsymbol{\phi}$, *and* $f(\cdot)$ is the activation function.

Noisy Threshold Model

- We consider the generalized linear model:

  $p(t = 1|a) = f(a)$, $a = \boldsymbol{w}^T\boldsymbol{\phi}$, and $f(\cdot)$ is the activation function.

- To motivate an alternative choice for the link function $f^{-1}(\cdot)$, we consider a noisy threshold model, as follows. For each input $\boldsymbol{\phi}_n$, we evaluate $a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n$ and then we set the target value according to

$$\begin{cases} t_n = 1 \text{ if } a_n \geq \theta \\ t_n = 0 \text{ otherwise} \end{cases}$$

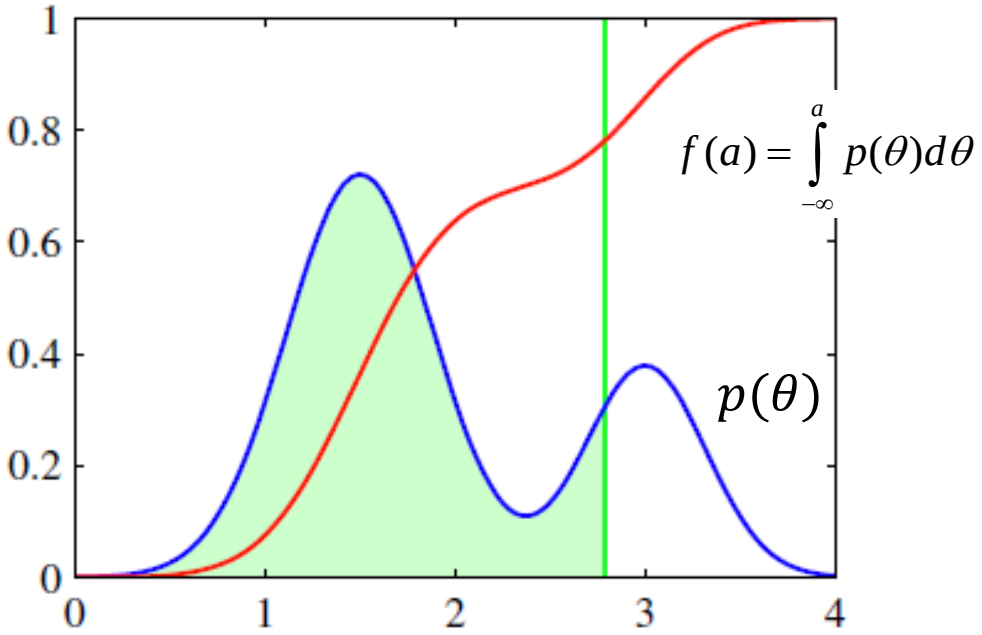
- *The threshold θ is stochastic and drawn from $p(\theta)$,* the corresponding activation function will be given by the cumulative distribution function

$$f(a) = \int_{-\infty}^{a} p(\theta)d\theta$$

$$p(t=1|\boldsymbol{\phi}) \equiv p\left(\theta \leq \underset{a}{\boldsymbol{w}^T\boldsymbol{\phi}}\right) = \int_{-\infty}^{a} p(\theta)d\theta = f(a)$$

Noisy Threshold Model

- Here $p(\theta)$ *is* a mixture of two Gaussians. Its cumulative distribution function $f(a)$ is shown.
- In the stochastic threshold model, the class label takes the value $t = 1$ if the value of $a = \boldsymbol{w}^T\boldsymbol{\phi}$ exceeds a threshold, otherwise it takes the value $t = 0$.
- This is equivalent to an activation function given by the cumulative distribution function $f(a)$.



Inverse Probit Function

- As a specific example, suppose that the density $p(\theta)$ is given by a zero mean, unit variance Gaussian. The corresponding cumulative distribution function is given by the inverse probit function:

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta \mid 0,1)\, d\theta$$

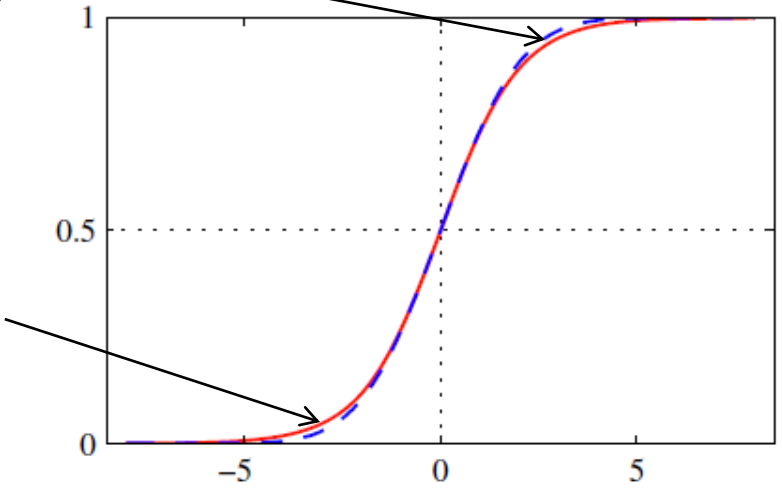
- It has a sinusoidal shape.

$$\Phi(\lambda a) = \int_{-\infty}^{\lambda a} \mathcal{N}(\theta \mid 0,1)\, d\theta$$

Scaled inverse Probit function

$$\sigma(a) = \frac{1}{1+e^{-a}}$$

Logistic sigmoid



MatLab Code

Probit Regression

- Note that the use of a more general Gaussian distribution does not change the model because this is equivalent to a re-scaling of the linear coefficients $\boldsymbol{w}$. Also note the relation:

$$erf(a) = \frac{2}{\sqrt{\pi}} \int_0^a e^{-\theta^2} d\theta \text{ (error function)}$$

$$\Phi(a) = \int_{-\infty}^{a} \mathcal{N}(\theta \mid 0,1) d\theta \qquad \Phi(a) = \frac{1}{2}\left\{1 + erf\left(\frac{a}{\sqrt{2}}\right)\right\}$$

- The generalized linear model based on a probit activation function is known as probit regression.
- We can determine the parameters of this model using MLE.
- The results using probit regression are similar to those of logistic regression.

Outliers

- Outliers can arise through errors in measuring the input $\boldsymbol{x}$ or through mislabeling of the target value t.
- Such points can lie a long way to the wrong side of the decision boundary seriously distorting the classifier.
- The logistic & probit models behave differently to outliers:
  - the tails of the logistic sigmoid decays asymptoticall $\sim \exp(-x)$ for $x \to \infty$, whereas
  - for the probit activation function they decay $\sim \exp(-x^2)$.
- Thus the probit model is more sensitive to outliers.
- Mislabelling target values can easily be incorporated as follows:

$$p(t \mid \boldsymbol{\phi}) = (1-\varepsilon)\, y(\boldsymbol{\phi}) + \varepsilon\left(1 - y(\boldsymbol{\phi})\right),\ y(\boldsymbol{\phi}) = activation\ \ function$$

- ϵ is the probability of mislabelling that can be set a priori or treated as hyperparameter to be inferred by the data.

- Opper, M. and O. Winther (2000a). Gaussian processes and SVM: mean field theory and leave-one-out. In A. J. Smola, P. L. Bartlett, B. Scholkopf, and D. Shuurmans (Eds.), *Advances in Large Margin Classifiers*, pp. 311–326. MIT Press.

IRLS for Probit Regression

- To apply the IRLS iterative algorithm, we need to compute the gradient and Hessian of the log likelihood.
- The gradient is computed easily noting that: $\nabla a_n = \boldsymbol{\phi}_n$ and

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}\,|\,\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln\left(1-y_n\right)\right\} \Rightarrow \frac{\partial E}{\partial y_n} = \frac{y_n - t_n}{y_n(1-y_n)}$$

- Also from $\Phi(a) = \frac{1}{2}\left\{1 + erf\left(\frac{a}{\sqrt{2}}\right)\right\}$ and $erf(a) = \frac{2}{\sqrt{\pi}}\int_{-\infty}^{a} e^{-\theta^2}\,d\theta \Rightarrow$

$$\frac{\partial y_n}{\partial a_n} = \frac{\partial \Phi(a_n)}{\partial a_n} = \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2}$$

- Thus the gradient is given as:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \frac{\partial E}{\partial y_n}\frac{\partial y_n}{\partial a_n}\nabla a_n = \sum_{n=1}^{N} \frac{y_n - t_n}{y_n(1-y_n)}\frac{1}{\sqrt{2\pi}} e^{-a_n^2/2}\boldsymbol{\phi}_n$$

IRLS for Probit Regression

- The computation of the Hessian is also straightforward. Note

$$\frac{\partial^2 E}{\partial y_n^2} = \frac{\partial}{\partial y_n}\frac{y_n - t_n}{y_n(1-y_n)} = \frac{y_n^2 + t_n - 2y_n t_n}{y_n^2(1-y_n)^2}$$

- This leads to:

$$\nabla E(\boldsymbol{w}) = \sum_{n=1}^{N} \frac{y_n - t_n}{y_n(1-y_n)} \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2} \boldsymbol{\phi}_n \Rightarrow$$

$$\nabla^2 E(\boldsymbol{w}) = \sum_{n=1}^{N} \left[\frac{\partial}{\partial y_n}\left\{\frac{y_n - t_n}{y_n(1-y_n)}\right\} \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2} \boldsymbol{\phi}_n \nabla y_n + \frac{y_n - t_n}{y_n(1-y_n)} \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2} (-a_n) \boldsymbol{\phi}_n \nabla a_n \right] =$$

($\nabla y_n = \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2} \boldsymbol{\phi}_n^T$, $\nabla a_n = \boldsymbol{\phi}_n^T$)

$$= \sum_{n=1}^{N} \left[\frac{y_n^2 + t_n - 2y_n t_n}{y_n(1-y_n)} \frac{1}{\sqrt{2\pi}} e^{-a_n^2/2} - a_n(y_n - t_n) \right] e^{-a_n^2/2} \frac{1}{\sqrt{2\pi}} \frac{1}{y_n(1-y_n)} \boldsymbol{\phi}_n \boldsymbol{\phi}_n^T$$

Generalizing our Results:

(1) $p(t|\phi(x)) = f(w^T \phi(x))$ from the Exponential Family, with

(2) f being the Canonical Link Function

Canonical Link Functions

- For the linear regression model with a Gaussian noise, the error function, corresponding to the negative log likelihood, is

$$E_D(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_n)\right)^2$$

- If we take the derivative wrt $\boldsymbol{w}$ of the contribution to the error function from a data point n, this takes the form of the 'error' $y_n - t_n$ times the feature vector $\boldsymbol{\phi_n}$, where $y_n = \boldsymbol{w^T}\,\boldsymbol{\phi_n}$.

- Similarly, for the combination of the logistic sigmoid activation function and the cross-entropy error function

$$E(\boldsymbol{w}) = -\ln p(\boldsymbol{t}\,|\,\boldsymbol{w}) = -\sum_{n=1}^{N}\left\{t_n \ln y_n + (1-t_n)\ln\left(1-y_n\right)\right\},$$

and for the softmax activation function with the multiclass cross-entropy error function

$$E(\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\ln p(\boldsymbol{T}\,|\,\boldsymbol{w}_1,\ldots,\boldsymbol{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

we again obtain this same simple form.

Canonical Link Functions

- This is a general result: Assume a conditional distribution for the target variable from the exponential family, along with a corresponding choice for the activation function known as the canonical link function.
- We again make use of the restricted form of exponential family distributions.

$$p(\boldsymbol{x} \mid \boldsymbol{\lambda}_k, s) = \frac{1}{s} h(\frac{1}{s}\boldsymbol{x}) g(\boldsymbol{\lambda}_k) \exp\left\{\frac{1}{s}\boldsymbol{\lambda}_k^T \boldsymbol{x}\right\}$$

- Note that here we are applying the assumption of exponential family distribution to the target variable t, in contrast to earlier discussion where we applied it to the input vector $\boldsymbol{x}$.
- We thus consider conditional distributions of the target variable t of the form

$$p(t \mid \eta, s) = \frac{1}{s} h(\frac{t}{s}) g(\eta) \exp\left\{\frac{\eta t}{s}\right\}$$

Canonical Link Functions

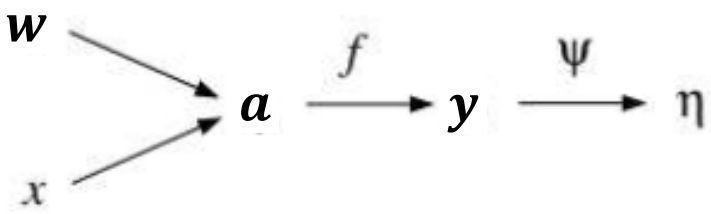
$$p(t \mid \eta, s) = \frac{1}{s} h(\frac{t}{s}) g(\eta) \exp\left\{ \frac{\eta t}{s} \right\}$$

- We will prove in a follow up lecture on generalized linear models that:

$$-\nabla \ln g(\eta) = \mathbb{E}\left[u(\boldsymbol{x})\right] \text{ for } p(\boldsymbol{x} \mid \boldsymbol{\eta}) = h(\boldsymbol{x}) g(\boldsymbol{\eta}) \exp(\boldsymbol{\eta}^T u(\boldsymbol{x}))$$

we see that the conditional mean of t, which we denote by y, is

$$y \equiv \mathbb{E}\left[t \mid \eta\right] = -s \frac{d}{d\eta} \ln g(\eta)$$



- Thus y and η must be related: $\eta = \psi(y)$
- We define a generalized linear model to be one for which y *is a* nonlinear function of a linear combination of the input (or feature) variables so that $y = f(\boldsymbol{w}^T \boldsymbol{\phi})$.
- $f(\cdot)$ is the activation function, and $f^{-1}(\cdot)$ is known as the link function in statistics.

- Nelder, J. A. and R.W. M.Wedderburn (1972). Generalized linear models. *Journal of the Royal Statistical Society, A* **135**, 370–384.

Canonical Link Functions

- Now consider the log likelihood for this model (as a function of η):

$$\ln p(t \mid \eta, s) = \sum_{n=1}^{N} p(t_n \mid \eta, s) = \sum_{n=1}^{N} \left(\ln g(\eta_n) + \frac{\eta_n t_n}{s} \right) + const$$

  where we are assuming that s is independent of η, *i.e.* all observations share a common scale parameter (which for a Gaussian distribution corresponds to the noise variance).

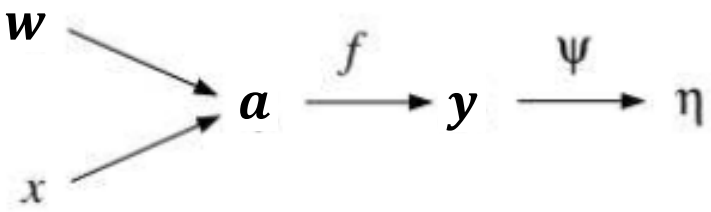
- The derivative of the log likelihood with respect to the model parameters $\boldsymbol{w}$ is then given by

$$\nabla_{\boldsymbol{w}} \ln p(t \mid \eta, s) = \sum_{n=1}^{N} \left(\frac{d}{d\eta_n} \ln g(\eta_n) + \frac{t_n}{s} \right) \frac{d\eta_n}{dy_n} \frac{dy_n}{da_n} \nabla a_n = \frac{1}{s} \sum_{n=1}^{N} (t_n - y_n) \psi'(y_n) f'(a_n) \boldsymbol{\phi}_n$$

$$where: \; a_n = \boldsymbol{w}^T \boldsymbol{\phi}_n, \;\; y_n = f(a_n), \;\; y = \mathbb{E}[t \mid \eta] = -s \frac{d}{d\eta} \ln g(\eta) \; (\Rightarrow \eta = \psi(y))$$

- There is considerable simplification if we take

$$f^{-1}(y) = \psi(y) \Rightarrow f(\psi(y)) = y \Rightarrow \psi'(y) f'(\psi) = 1$$



Canonical Link Functions

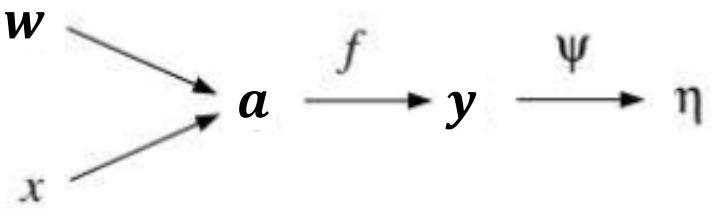
$$\nabla_w \ln p(\boldsymbol{t} \mid \eta, s) = \frac{1}{s}\sum_{n=1}^{N}(t_n - y_n)\psi'(y_n) f'(a_n)\boldsymbol{\phi}_n$$

$$where:\ a_n = \boldsymbol{w}^T\boldsymbol{\phi}_n,\ \ y_n = f(a_n),\ \ y = \mathbb{E}[t \mid \eta] = -s\frac{d}{d\eta}\ln g(\eta)$$

- There is considerable simplification if we take

$$f^{-1}(y) = \psi(y) \Rightarrow f(\psi(y)) = y \Rightarrow \psi'(y) f'(\psi) = 1$$

- Also because



$$f^{-1}(y) = \psi(y) \Rightarrow a = \eta \Rightarrow \begin{matrix} a = \psi(y) \\ Also\ use: \psi'(y) f'(\psi) = 1 \end{matrix} \Rightarrow \psi'(y) f'(a) = 1$$

- The gradient of the error is now simplified to a familiar form as:

$$\nabla E(\boldsymbol{w}) = \frac{1}{s}\sum_{n=1}^{N}(t_n - y_n)\boldsymbol{\phi}_n, \qquad \begin{matrix} \text{Gaussian model}: s = \beta^{-1} \\ \text{Logistic model}: s = 1 \end{matrix}$$